Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Desarrollo de una aplicación móvil en el ámbito deportivo

Autor: Benjamín Lee Chong García

Tutor: David Griol Barres

Leganés, septiembre de 2016

Título: **Desarrollo de una aplicación móvil en el ámbito deportivo**

Autor: **Benjamín Lee Chong García**

Director: **David Griol Barres**

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Trabajo Fin de Grado el día __ de _____ de 20__ en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Agradecimientos

En primer lugar quiero agradecer a mi tutor David Griol por toda la confianza depositada, por el apoyo, los consejos y la paciencia que ha tenido durante el trascurso del proyecto.

En segundo lugar agradecer a mi familia por el apoyo recibido, por estar allí en los momentos más difíciles y por animarme a finalizar este trabajo. Quiero agradecer especialmente a mi madre por todo el sacrificio y todas las oportunidades que me ha brindado a lo largo de mi vida, sin ellas no hubiera podido llegar donde estoy ahora.

Quiero dar un agradecimiento muy especial a Tess por su paciencia, comprensión y por estar siempre a mi lado cuando lo necesitaba. Agradecer los ánimos y consejos que me ha dado y que me han ayudado a seguir en el camino para acabar este proyecto.

Agradecimientos a todos mis profesores de la carrera. Me han ayudado a ser mejor y a esforzarme más para llegar a ser un buen profesional.

También quiero agradecer a mis compañeros de carrera, especialmente a todas aquellas personas con las que he compartido prácticas, trabajos, dolores de cabeza y también tardes más relajadas. Quisiera agradecer a Edu por ser un gran compañero y amigo durante esos primeros años de carrera.

Por último quiero agradecer a mis amigos por el apoyo y consejos recibidos, especialmente a mis compañeros de equipo que me han estado ayudando con ideas y sugerencias para crear una buena aplicación.

Resumen

La finalidad para el desarrollo de una aplicación deportiva es crear un nuevo entorno y experiencia para personas que tienen dificultades cuando utilizan aplicaciones tradicionales de deporte. Para ello, proporciona características y funcionalidades específicas que permiten interactuar con la aplicación de forma no convencional: por medio de la voz o por medio del entorno que lo rodea.

Toda la información que muestra la aplicación es acerca del deporte Ultimate Frisbee, de los aspectos más importantes de este deporte: las reglas, los equipos y los torneos que existen. Por este motivo, la aplicación creada utiliza las distintas salidas de los dispositivos móviles para mostrar los resultados de las acciones del usuario, y utiliza las entradas para obtener las consultas del mismo.

Las entradas del sistema pueden ser táctiles (teclado o por pantalla) u orales (reconocimiento y síntesis de voz), obteniendo como resultado una serie de informaciones que pueden ser visualizadas por pantalla o leídas por el dispositivo móvil.

Debido a la naturaleza del sistema, esta aplicación puede ser utilizada o accedida por personas con problemas de visión o aquellas personas con discapacidades motoras que no pueden utilizar los medios tradicionales.

Para proporcionar un entorno mucho más amigable se ha dotado a la aplicación de una serie de ayudas sonoras y visuales (botones) para la búsqueda de información, además de las preferencias de localización para la búsqueda automática de equipos o torneos.

Para aumentar la experiencia del usuario se ha proporcionado al sistema un entorno de realidad aumentada. Estas técnicas proporcionan visualización, selección y la ruta hacia los torneos y equipos de la zona geográfica donde se encuentre el usuario.

Por último el sistema está orientado para dispositivos móviles con el sistema operativo Android instalado, por lo que este documento incluye un estudio tanto el entorno de desarrollo (Android Studio), como de las tecnologías utilizadas en el transcurso del proyecto, además de la explicación de las funcionalidades y las futuras mejoras.

Palabras clave: Android, Dispositivos móviles, realidad aumentada, reconocimiento y síntesis de voz, Ultimate Frisbee.

Abstract

The purpose for the development of a sport application is create a new environment and new experiences for those people who have difficulty when they use traditional sport applications. For that purpose, the aplication provides specific features and functionalities that allow interact with the application in an unconventional way: by voice or by the environment around them.

All the information is about Ultimate Frisbee Sport, the most important aspects of the sport: rules, teams and tournaments. For that reason, the created application uses the output devices to show all the results of the user's actions, and uses the input devices to obtain the user's query.

The entries of the system may be tactiles (keyboard and screen) or you may use the voice (speech recognition and speech synthesis), and the results that you obtain can be displayed on the screen or can be read it by the mobile device.

Due to the nature of the system this application can be used or accessed by persons with visual problems or those people with motor impaired who can’t use the traditional media.

To give an friendly enviroment to users, the system has built with visuals and noise functions to help in searching information. Also, the system incorporates localization preference for the automatic search of teams or tournaments.

To increase the user experience, the system provide an environment of augmented reality. These techniques provide a visualization, selection, and a route to tournaments and teams in your geographical area.

Finally the system is geared for mobile devices with Android operating system. This document includes a study of the development environment (Android Studio), and the technologies used for its development; also, includes the explanation of the features and potential or future improvements.

Keywords: Android, mobile devices, augmented reality, speech recognition and synthesis, Ultimate Frisbee.

Índice general

Contenido

Índice de Figuras

Índice de Tablas

Capítulo 1

Introducción

1.1 Introducción

A lo largo de la historia el deporte ha permitido a las personas desarrollarse más, empatizar con otras, y ha sido nexo de unión entre seres humanos, ya sea motivándolos hacia una victoria o una meta determinada, o como apoyo hacia un atleta o atletas con los que comparten una afinidad.

Este motivo nos lleva a pensar que en la actualidad debemos potenciar el desarrollo de tecnologías que permitan acercar y practicar el deporte a las personas, sin importar su condición social, su raza, religión u otras distinciones.

Este el motivo principal por el que se ha escogido el desarrollo de una aplicación en el ámbito deportivo: poder acercar un deporte a personas que no pueden utilizar los medios tradicionales para interactuar con dispositivos móviles.

El deporte es una las actividades por las que el ser humano ha sido capaz de unirse más y los dispositivos móviles son herramientas que permiten acceder, casi instantáneamente, desde cualquier punto del planeta a cualquier tipo de información. La unión de estos dos mundos puede proporcionar una vía para que las personas se interesen por el deporte puedan saber en cualquier momento cuales son las novedades y eventos de un deporte que normalmente no tienen oportunidad de practicar.

En nuestro caso hemos desarrollado una aplicación de ámbito deportivo enfocado en el deporte Ultimate Frisbee.

La aplicación interactúa con los usuarios para mostrar los equipos, torneos y reglas que existen, y utiliza la cámara del dispositivo para marcar los puntos, por medio de marcadores, donde se encuentran los torneos y equipos más cercanos.

Como hemos apuntando anteriormente, la aplicación debe ser accesible para personas que tengan dificultades visuales, por ello, la interactividad entre el usuario y el dispositivo móvil se lleva a cabo por medio de las entradas táctiles u orales del dispositivo, y las salidas por medio de la pantalla y la salida de voz del dispositivo.

Al ser un sistema donde utilizamos varios canales de entrada y de salida para obtener y mostrar la información podemos decir que estamos en un sistema multimodal.

En primer lugar, la multimodalidad apunta a la variedad de modos o recursos semióticos utilizados para significar y que confluyen en un mismo evento comunicativo [1]. Cualquier texto que incluya más de un recurso que sirva para significar puede ser definido como multimodal independientemente del soporte en el cual se distribuya [2].

Un sistema multimodal en informática es aquel sistema informático (ordenador, dispositivos móviles, etc.) capaz de obtener información de los usuarios a través de los diferentes canales de comunicación humanos tales como la voz, los gestos y los movimientos. El sistema multimodal es además capaz de comprender la información dada y proporcionar una respuesta válida [3].

El uso de tecnologías y técnicas multimodales (como el reconocimiento voz) han ido incrementando a lo largo de las actualizaciones de estos dispositivos [4] pero no ha sido así el uso cotidiano o la posibilidad de su uso dentro de las aplicaciones.

El problema que conlleva la utilización de estos sistemas consiste en la inherente espontaneidad de los usuarios a la hora de realizar una consulta, la comprensión de la consulta por parte del dispositivo, y la obtención de una respuesta coherente a la pregunta dada.

Otra funcionalidad que debe aportar la aplicación es la visualización de torneos y equipos cercanos, por medio de la cámara del dispositivo. Esta funcionalidad debe realizarse con técnicas de Realidad Aumentada.

La Realidad Aumentada son todas aquellas tecnologías que permiten la superposición, en tiempo real, de imágenes, marcadores o información generados virtualmente, sobre imágenes del mundo real [5]. Es la fusión de la realidad con elementos virtuales para enriquecer la percepción de la realidad.

Los dispositivos móviles cada vez están mejor equipados para soportar este tipo de tecnología, y cada vez es más frecuente su uso: Pokemos Go y tripAdvisor

Pokemos Go

Es un juego de una serie de dibujos animados muy conocido, y consiste en buscar criaturas llamadas *Pokémons* con la ayuda del GPS del móvil, para después capturarlas.

Este juego muestra en la pantalla del dispositivo al jugador, situándolo geográficamente en el mapa, y mostrando los puntos donde las criaturas se encuentran.

Figura 1: Pokémon Go interfaz

Es un juego de una serie de dibujos animados muy conocido, y consiste en buscar criaturas llamadas *Pokémons* con la ayuda del GPS del móvil, para después capturarlas.

Este juego muestra en la pantalla del dispositivo al jugador, situándolo geográficamente en el mapa, y mostrando los puntos donde las criaturas se encuentran [6].

TripAdvisor

Aplicación de una conocida página de búsqueda de restaurantes y hoteles. Utiliza la cámara del dispositivo para mostrar los hoteles, restaurantes o lugares de interés más cercarnos.

Figura 2: TripAdvisor interfaz

Implementamos está tecnología para hacer más atractiva la aplicación y para añadir una experiencia diferente a los usuarios que la utilicen. De esta forma, los usuarios se encuentran con una aplicación útil y novedosa, que permite interactuar con el entorno que los rodea [7].

1.2 Deporte. Ultimate Frisbee

Una vez definidos que es un sistema multimodal vamos a abarcar el contexto de nuestra aplicación: Ultimate Frisbee.

El Ultimate Frisbee es un deporte de no contacto, auto arbitrado, mezcla de rugby, fútbol, baloncesto y otras disciplinas, que se juega con un disco en un campo de dimensiones definidas, con una caracterización por encima de otras: el espíritu de juego, es decir, el juego limpio y el respeto al rival [8].

Uno de los características principales de este juego es la participación personas de diferentes ámbitos sin distinción de género, raza o religión. Es un juego integrador donde hombres y mujeres pueden jugar al mismo tiempo, es decir, es un deporte mixto.

La motivación para elegir este deporte es que al ser precisamente un deporte tan abierto no existe una aplicación móvil lo suficientemente amplia que abarque todas sus características y que además sea accesible y amigable para personas con dificultades visuales. De esta forma podemos, por una parte, facilitar el contenido y los eventos del deporte tanto a personas con inclinaciones deportivas, indistintamente si tienen dificultades visuales o no, y por otra, atraer a aquellas personas que no les interese tanto el deporte pero quieran practicar uno nuevo.

Para desarrollar la aplicación se han estudiado algunas aplicaciones que existe en la actualidad, y la aplicación de sistemas de realidad aumentada y de ayuda o estímulos sonoros.

En el mundo de las aplicaciones móviles existe un sinfín de aplicaciones destinadas al uso deportivo [9], al seguimiento de equipos y a la localización de eventos deportivos, pero muy pocas aplicaciones están destinadas a personas que sufren alguna discapacidad, ya que todas utilizan medios tradicionales para interactuar con el sistema, por lo que impide su utilización a personas con dificultades visuales y sonoras.

1.3 Objetivos

El objetivo principal es el desarrollo de una aplicación deportiva para dispositivos móviles Android, que disponga de una interfaz multimodal para facilitar su uso y que integre funcionalidades de realidad aumentada para la búsqueda de los diferentes eventos deportivos.

Por lo tanto, podemos dividir los componentes fundamentales de la aplicación en tres módulos: módulo base, módulo asistente oral y módulo realidad aumentada.

Los módulos se fundamentan en un estudio de las tecnologías más apropiadas para desarrollarlos. Además cada módulo se unirá, integrará y comunicará con el proyecto principal, de esta forma el resultado final será una unidad completa operable.

Módulo base

El módulo base consiste en la representación de todos los aspectos esenciales de la aplicación, tales como el inicio de sesión, las preferencias del usuario, la búsqueda y representación de las reglas, los equipos y los torneos.

Este módulo está a su vez compuesto de otros módulos internos que deberán comunicarse entre sí, además de permitir al usuario las siguientes funciones:

- Registro e inicio de sesión.
- Listado de torneos, equipos y reglas favoritas
- Búsqueda de información en las reglas.
- Búsqueda de equipos y torneos.
- Cambio en las preferencias de usuario.

Este módulo es el encargado de comunicarse con los servidores externos creados, con la base de datos interna, y con las páginas web de donde buscaremos los equipos y los torneos que existen.

Debido a la complejidad del módulo se ha completado un estudio detallado de las siguientes tecnologías:

- Estudio de la tecnología Android, la plataforma y las distintas aplicaciones.
- Integración de bases de datos en Android.
- Estudio de las alternativas de servidores externos para crear, manipular y gestionar una arquitectura cliente-servidor.
- Estudio de la alternativas para el web Scraping.
- Estudio de paradigmas de diseño: Google Materials

Módulo asistente oral

El módulo asistente oral tiene como principal finalidad suministrar las herramientas necesarias para la comunicación mediante la voz entre el usuario y la aplicación.

Por ello, se ha completado un estudio de las siguientes tecnologías:

- Tecnologías de reconocimiento de voz.
- Tecnologías de síntesis de voz.

Este componente se ha integrado con el módulo base, ya que las funciones a desarrollar son muy similares al módulo citado. Esto quiere decir que el objetivo de este módulo es facilitar oralmente el inicio de sesión, el registro, la búsqueda de equipos, reglas y torneos favoritos y la elección o no selección de favoritos.

Módulo Realidad Aumentada

Por último el módulo de Realidad Aumentada tiene como objetivo final la visualización, por medio de la cámara del dispositivo móvil, de torneos y equipos más cercanos, además de presentar la ruta más cercana entre el punto inicial (localización del usuario) y el punto final (torneo o equipo elegido).

Se ha realizado el estudio de las siguientes tecnologías:

- Tecnologías de realidad aumentada.
- Tecnologías de mapas y rutas en Android.
- Implementación de tecnologías de realidad aumentada con entornos de desarrollos nativos Android.
- Estudio sobre las distintas aplicaciones que utilizan la realidad aumentada.

A diferencia de los otros dos módulos, este módulo no estará disponible para toda la aplicación, solo se visualizará o se representará para los sub-módulos torneos y equipos: tendrá que proporcionar la información necesaria de los equipos y los torneos cuando este sea llamado por el usuario.

Podemos diferenciar además dos sub-módulos: el propio de la realidad aumentada y la parte de mapas y rutas.

1.4 Fases de Desarrollo

Para la realización del presente proyecto se ha seguido el paradigma organizativo WBS *(*Work BreakdownStructure*)*, cuya finalidad es estructurar el trabajo de forma jerárquica, dividiendo el proyecto en 3 fases: Planificación, Ejecución y Documentación.

La Figura 3 representa el conjunto de fases desarrolladas a lo largo del proyecto

Fase 1: Planificación

- Estudio del sistema operativo Android: Estudio de las posibilidades que ofrece esta plataforma para la realización del proyecto: estudio de integración de bases de datos, comunicación de servidores y diseño. Además en este apartado veremos las distintas tecnologías que nos ayudarán a implementar nuestra aplicación como: SQLite, MySQL, PHP, Java, XML, JSON y las librerías Volley, JSoup y Retrofit.

- Estudio de las tecnologías de sistemas de diálogo: Aproximación a las arquitecturas de sistemas de diálogos, a los requisitos que hacen falta para su integración y las dificultades que conlleva.

- Estudio de las tecnologías de realidad aumentada: Aproximación a las arquitecturas de realidad aumentada, a los requisitos funcionales y de hardware que hacen falta para su integración y las dificultades que conlleva esta implantación.

- Definición de los requisitos funcionales: definición del funcionamiento y el flujo de acciones básicas de nuestra aplicación.

Fase 2: Ejecución

- Diseño base de la aplicación: diseño de los distintos módulos, las clases y la comunicación que existe entre los módulos y sus librerías.

- Desarrollo del código de la aplicación: programación de la aplicación en el entorno Android Studio.

- Integración de los distintos módulos: integración de las distintas partes de la aplicación para que actúe como una unidad.

- Pruebas y evaluación: realización de un número extenso de pruebas para su posterior evaluación.

Fase 3: Documentación

- Memoria del proyecto: Desarrollo y preparación de este documento.

- Presentación: Preparación de la presentación del proyecto.

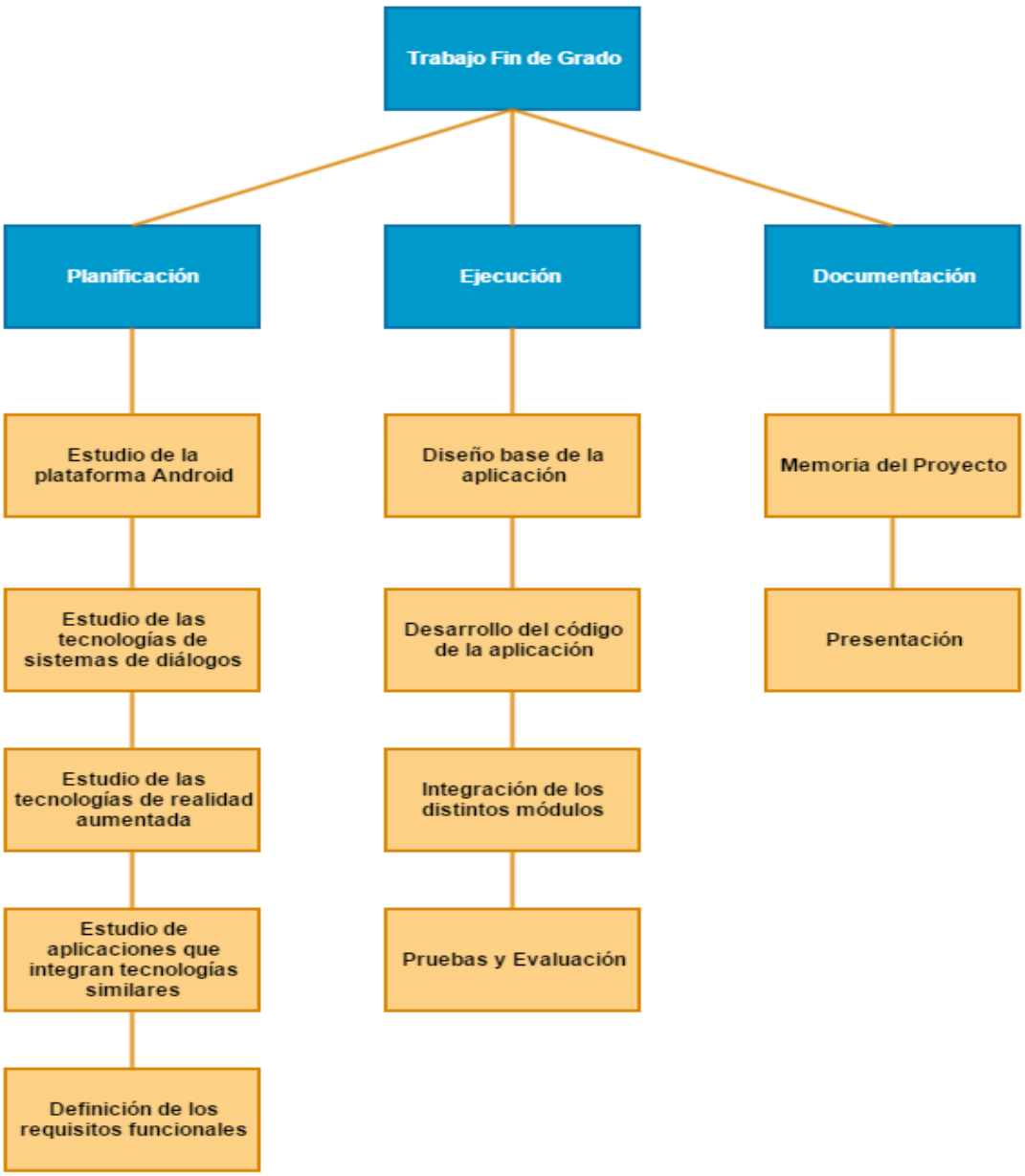


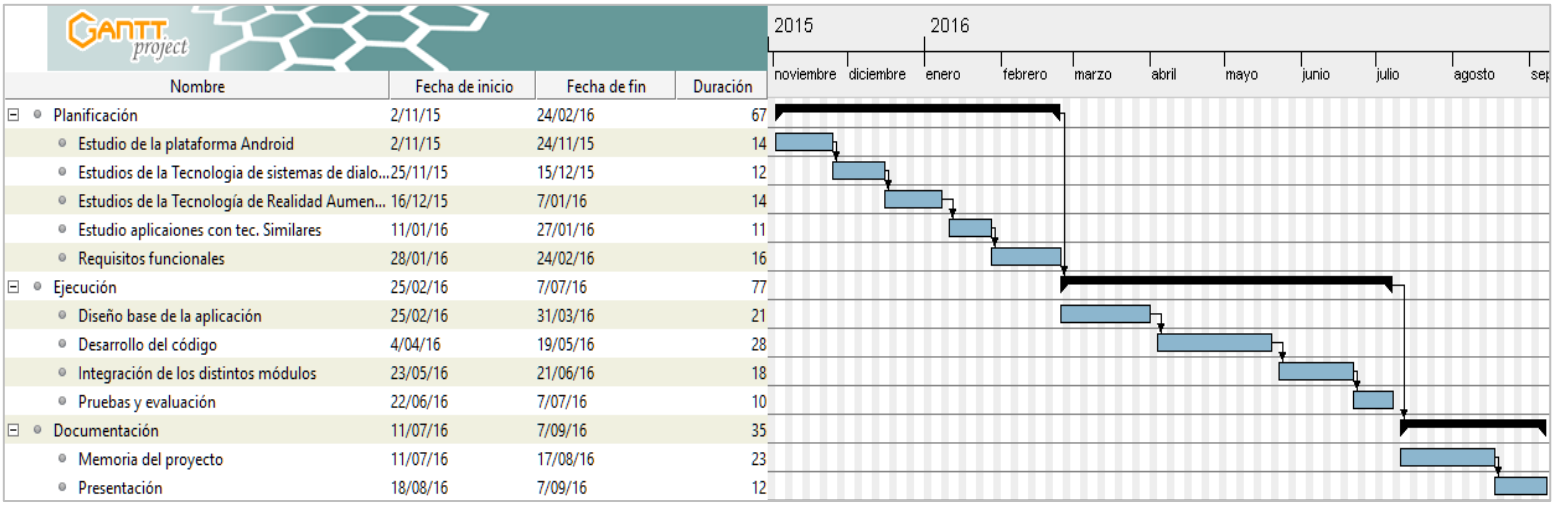
Figura 3: Diagrama de planificación para el desarrollo de la aplicación

1.5 Planificación temporal

Para realizar el seguimiento de las fases de desarrollo se ha utilizado la herramienta GanttProject que permite realizar diagramas de Gantt.

Los diagramas contienen la duración estimada de las fases de desarrollo, es decir, la estimación en tiempo de las tareas que debemos realizar en las fases de Planificación, Ejecución y Desarrollo.

La duración se estima con la premisa de trabajar 3 horas al día de lunes a jueves, empezando desde la fecha de inicio del proyecto hasta el final del mismo.



Nombre	Fecha de inicio	Fecha de fin	Duración
Planificación	2/11/15	24/02/16	67
Estudio de la plataforma Android	2/11/15	24/11/15	14
Estudios de la Tecnologia de sistemas de dialo...	25/11/15	15/12/15	12
Estudios de la Tecnología de Realidad Aumen...	16/12/15	7/01/16	14
Estudio aplicaiones con tec. Similares	11/01/16	27/01/16	11
Requisitos funcionales	28/01/16	24/02/16	16
Ejecución	25/02/16	7/07/16	77
Diseño base de la aplicación	25/02/16	31/03/16	21
Desarrollo del código	4/04/16	19/05/16	28
Integración de los distintos módulos	23/05/16	21/06/16	18
Pruebas y evaluación	22/06/16	7/07/16	10
Documentación	11/07/16	7/09/16	35
Memoria del proyecto	11/07/16	17/08/16	23
Presentación	18/08/16	7/09/16	12

Figura 4: Diagrama de Gantt con la planificación temporal del proyecto

1.6 Estimación de recursos del proyecto

Los recursos empleados para la realización del trabajo fin de grado son los siguientes:

1.6.1 Recursos Hardware

- Ordenador sobremesa HP.
- Ordenador portátil ASUS.
- Movil HTC one.
- Tablet Nvidia Shiedl.
- Cables usb.

1.6.2 Recursos Software

- Entorno nativo de Android: Android Studio.
- JVM (Java Virtual Machine) para el desarrollo de código JAVA.
- JDK (Java development kit) kit de desarrollo en JAVA.
- XAMPP: sistema de gestión de base de datos en un servidor local.

- Librería Retrofit: Librería que permite la comunicación con el servidor REST local por medio de operaciones POST, para la obtención de datos de la base de datos local.

- Librería Volley: Librería que permite llamadas a recursos online. La utilizaremos para obtener el contenido de las consultas a la página WEB que contiene la información de los equipos y los torneos.

- Libreía JSOUP: Librería que nos permite estructurar (parsear) la información obtenida por la librería Volley y convertirla en objetos que son fácilmente manipulables.

- Librería TextToSpeech Android: Librería que ofrece la tecnología necesaria para sintetizar textos y su inmediata reproducción sonora por medio del altavoz del dispositivo móvil.

- Librería Speech Android: Nos permite reconocer el audio entrante y manipularlo con el propósito de obtener un resultado.

- Accesibilidad de Android: Herramienta desarrollada por Google para personas con dificultades visuales, y que permite navegar por el sistema Android con ayuda de un asistente sonoro.

- Wikitude: herramienta desarrollada para implementar entornos de realidad aumentada utilizando diversas tecnologías tales como: JavaScript, CSS, HTML o Java.

- Google maps: librería que permite mostrar mapas y rutas en nuestro dispositivo móvil.

- Notepad++: editor de texto que nos permite trabajar en múltiples ventanas y tipos de archivos.

- GanttProject: Herramienta utilizada para crear los diagramas de Gantt.

- Google Drive: Herramienta para alojar múltiples tipos de archivos. En nuestro caso guardará las versiones del proyecto así como las versiones de la documentación.

- UML: Herramienta para desarrollar las diferentes ilustraciones de nuestro proyecto.

- Adobe Photoshop: Herramienta de edición de imagen para la manipulación de nuestro logo.

- Adobe Illustrator: Herramienta para la creación de nuestro logo.

1.7 Estructura de la memoria del proyecto

En este apartado describiremos como se estructura este documento:

Capítulo 1 Introducción: Este capítulo incluye los propósitos, motivaciones y objetivos que han inducido a la realización de este proyecto, además de presentar las tecnologías, recursos y la planificación del proyecto en sí.

Capítulo 2 Estado del Arte: Presenta un exposición detallada de los sistemas de Android, así como de las tecnologías de reconocimiento y síntesis de voz y de realidad aumentada. Las alternativas detalladas en este capítulo describen el funcionamiento e integración de las mismas en aplicaciones Android.

Capítulo 3 Análisis del sistema: En este apartado se describe una visión general de sistema, su arquitectura y las tecnologías que se utilizan, además de presentar los requisitos funcionales, no funcionales del sistema y los casos de uso.

Capítulo 4 Diseño y descripción de los módulos del sistema: Se detalla el flujo de datos de cada módulo, y como se comunican las distintas partes de la aplicación. Se presentan varios escenarios posibles para complementar los requisitos funcionales. En este capítulo además se describirá el resultado final de la aplicación.

Capítulo 5 Evaluación: Se realiza una serie de evaluaciones de la aplicación, realizando un test sobre la misma. De esta forma se obtendrá el grado de satisfacción de los usuarios y se verificará si los objetivos propuestos en el capítulo 1 se cumplen.

Capítulo 6 Mejoras Futuras y conclusiones: Se expondrá las posibles mejoras de la aplicación y las conclusiones finales del proyecto.

Capítulo 7 Presupuesto: Presupuesto detallado del desarrollo de la aplicación.

Capítulo 8 Definiciones y abreviaturas: Lista detallada con las definiciones de los términos y abreviaturas aparecidas en este documento.

Capítulo 9 Bibliografía: Lista de las referencias bibliográficas utilizadas a lo largo del proyecto.

Capítulo 2

Estado del arte

En este capítulo vamos a abordar las diferentes tecnologías que existen para el desarrollo de la aplicación, el estado actual en las que se encuentran y un análisis del grado de implementación de estas tecnologías en los entornos de desarrollo Android.

Para empezar hablaremos del sistema operativo para el que va dirigido la aplicación (Android) para seguir después con el estudio de los sistemas de diálogo y finalizar con los entornos de Realidad Aumentada.

2.1 Android Studio

En este apartado analizaremos el entorno de desarrollo Android Studio. Entorno donde vamos desarrollar la aplicación, por lo que explicaremos sus características más importantes.

2.1.1 Android

Android es un sistema operativo basado en el núcleo de Linux diseñado inicialmente por Android Inc., y comprada posteriormente en 2005 por Google [10].

Uno de los atractivos del sistema operativo es que está basado en código libre, es decir, cualquier programador que se interese por este sistema puede desarrollar una aplicación desde cero, en su casa y con ningún coste adicional. Además está orientado para el uso en dispositivos móviles como smartphones, tablets, relojes inteligentes, televisores y automóviles.

Estas características unidas a las numerosas actualizaciones que han surgido por parte de los desarrolladores, y a la extensa documentación que soporta, han hecho de Android unas de los sistemas más populares, llegando a alcanzar una cuota de mercado casi del 83% en 2015 (Tabla 1) [11]; lo que hace de Android el sistema más extendido entre las plataformas móviles.

Sistema operativo	Unidades 2016	Market Share 2016 (%)	Unidades 2015	Market Share 2015 (%)
Android	296,912.8	86.2	271,647.0	82.2
iOS	44,395	12.9	48,085.5	14.6
Windows	1,971.0	0.6	8,198.2	2.5
Blackberry	400.4	0.1	1,153.2	0.3
others	680.6	0.2	1,229.0	0.4
Total	**344,359.7**	**100.0**	**330,312.9**	**100.0**

Tabla 1: Cuota de mercado último trimestre fiscal del año 2016

Por su parte dentro de las diferentes actualizaciones del sistema operativo podemos observar que existe una distribución variada entre las distintas versiones (Figura 5), lo que demuestra el gran problema que supone programar con esta tecnología [12].

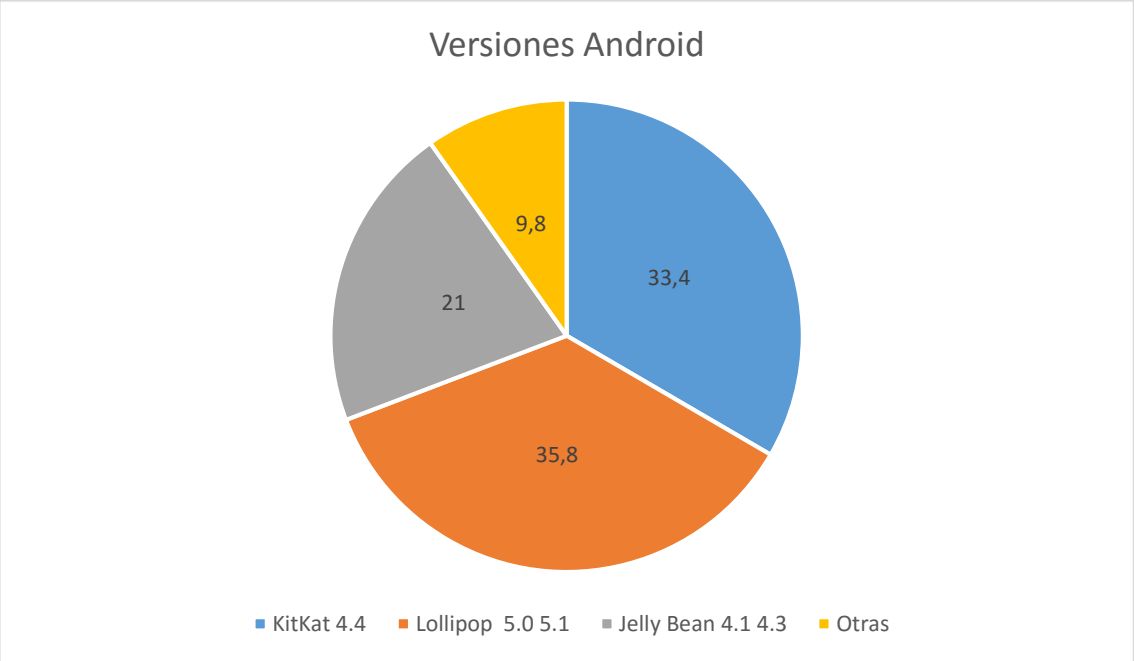


Figura 5: Distribución de las versiones Android

El problema de la fragmentación del sistema supone para el programador elegir entre las distintas versiones y razonar cuál de ellas le es más favorable.

En el presente proyecto se ha decidido utilizar la versión 5.0 (Lollipop) de Android porque nos permite realizar y testear las siguientes características [13]:

- Utilización del paradigma de diseño Google Materials
- Utilización de widget ReciclerView

- Compatibilidad con OpenGL: librerías gráficas. Permite desarrollar aplicaciones que combinen gráficos 2d y 3D

- Mejorar el rendimiento de la batería.

- Soporte para dispositivos de 64 bits [14]

- Mejoras en el sistema de habla e introducción de: “OK Google”

- Mejoras en los hilos multitarea.

Por otra parte mantiene las siguientes características [15]:

- Diseño: Plataforma adaptable a pantallas de grandes resoluciones, bilbioteca de gráficos 2D y 3D, basadas en compatibilidades OpenGL

- Almacenamiento: Base de datos interna y ligera: SQLite

- Conectividad: Tecnologías actuales soportadas: GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth, Wi-Fi, LTE, HSDPA, HSPA+, NFC y WiMAX, GPRS, UMTS y HSDPA+

- Mensajería: permite la utilización de envío y recepción de mensajes SMS y MMS.

- Navegador web: Por defecto navegador basado en WebKit emparejado com JavaScript

- Soporte de Java: En la versión 5.0 se utiliza Android Runtime (ART) como entorno de ejecución de Java.

- Soporte Multimedia: WebM, H.263, H.264 (en 3GP o MP4), MPEG-4 SP, AMR, AMR-WB (en un contenedor 3GP), AAC, HE-AAC (en contenedores MP4 o 3GP), MP3, MIDI, Ogg Vorbis, WAV, JPEG, PNG, GIF y BMP.

- Streaming: 3GPP PSS, ISMA; descarga progresiva de HTML5 con soporte Flash.

- Hardware adicional: soporte de cámaras de fotos, de vídeo, pantallas táctiles, GPS, acelerómetros, giroscopios, magnetómetros, sensores de proximidad y de presión, sensores de luz, gamepad, termómetro, aceleración por GPU 2D y 3D.

- Aplicaciones: Google Play: Catálogo extenso de aplicaciones de diversa índole.

- Otras características: Pantalla multi-tactil; videollamada, multitarea; características basadas en voz, tethering.

2.1.2 Entorno de desarrollo Android Studio, Android SDK

El entorno de desarrollo de aplicaciones en Android se le conoce como Android Studio o Android SDK, desarrollado por Google y basado en el software IntelliJ IDEA de JetBrains. Nos permite realizar aplicaciones Android respetando las características que debe poseer una aplicación de esta naturaleza, y teniendo en cuenta la Arquitectura de estos dispositivos (Figura 6).

Por otra parte este entorno nos permite: la edición de código de primer nivel, nos permite depurar código in situ, y nos proporciona herramientas de rendimiento para testear nuestra aplicación [16] (Figura 7).

Características importantes del SDK de Android:

- Utiliza Java y XML como lenguajes principales.
- Las mencionadas herramientas de rendimiento.
- Visualización de las pantallas finales
- Depuración de código.
- Compilación basada en Gradle.
- Emulador rápido
- Entorno unificado para el testeo de aplicaciones en diferentes entornos Android.
- Instant Run que permite introducir cambios cuando se ejecuta la aplicación.
- Integración de plantillas de código y GitHub.
- Gran cantidad de herramientas y frameworks de prueba.
- Diseñado para soportar compatibilidades con C++ y NDk.
- Facilitación de Google Cloud Messaging y App Engine debido al soporte integrado de Google Cloud Platform.

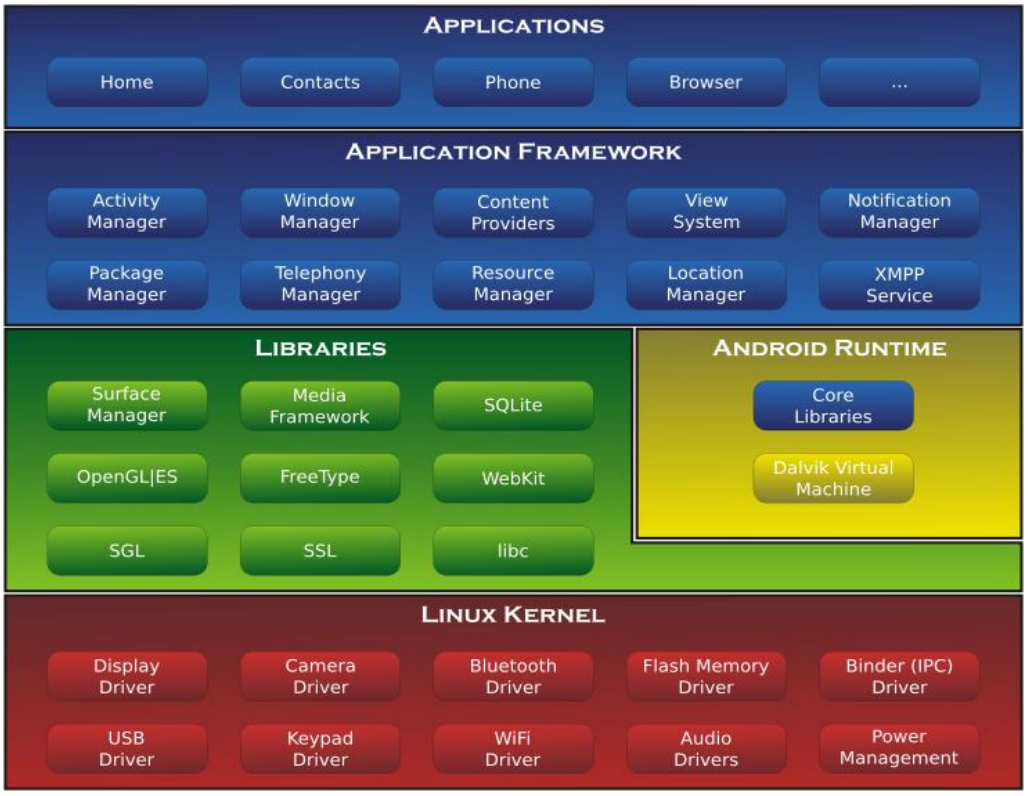


Figura 6: Arquitectura Android

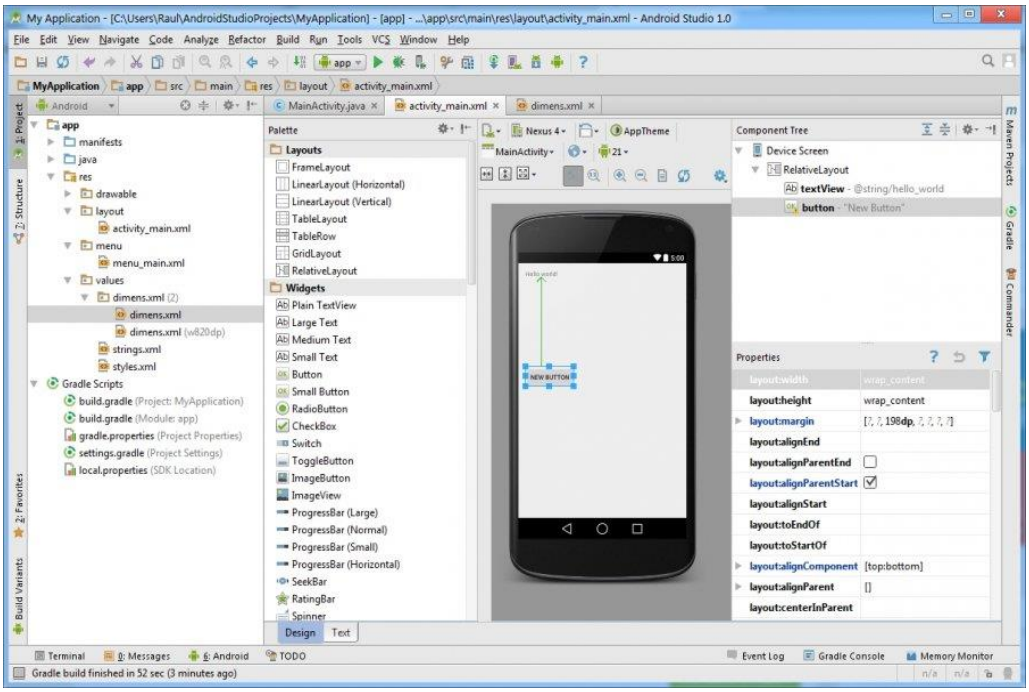


Figura 7: Android Studio

2.2 Servidores externos a la aplicación

Para la realización de nuestra aplicación es necesario la utilización de servidores externos para dar soporte a la base de datos interna del dispositivo, de esta forma todos los cambios que se guarden serán registrados y podrán se accedidos por distintos dispositivos móviles con un mismo usuario.

Vamos a ver qué posibilidades nos ofrecen las distintas tecnologías que existen:

2.2.1 Xampp

Es una herramienta que nos proporciona servicios web Apache y gestión de base de datos MySQl. Se instala como servidor local y es utilizado para testear aplicaciones web y también como un servidor Web [17].

Esta herramienta puede utilizar PHP, Perl además de las tecnologías web habituales (HTML, HTML5, CSS, Jquery, etc) para la creación de páginas o contenido web.

La usabilidad de esta herramienta y la flexibilidad que supone poder tener el servidor y la base de datos en local, además de poder implementar servicios REST, hace de esta herramienta una alternativa a considerar.

2.2.2 APPserv

Herramienta similar a Xampp proporciona un servicio web Apache que funciona que incorpora MySQL y PHP [18].

La ventaja principal de esta herramienta es que nos permite instalar o desinstalar elementos que no vamos a necesitar, lo que es conveniente cuando ya tenemos instalada una base de datos local.

La principal desventaja es que no tiene una interfaz de administración.

2.2.3 Servidores no locales

Tenemos muchas alternativas cuando tratamos de alojar nuestros datos o páginas web en servidores no locales: desde alojamientos gratuitos hasta de pago, dependiendo de nuestras necesidades.

Los alojamientos o Hosting normalmente ofrecen servicios o cuotas gratuitas para alojar nuestra información, sin embargo, estos servicios son limitados o duran un tiempo

determinado, por lo que si el proyecto o la aplicación son muy grandes o necesitan alojar muchos datos la alternativa que nos queda es la opción de pago.

Por otro lado tenemos los servicios *cloud computing* [19].que no son más que máquinas virtuales que actúan conjuntamente para ofrecernos almacenamiento en red.

La gran ventaja de un servicio cloud es la gran flexibilidad y la escalabilidad que estas poseen frente al Hosting tradicional; sin embargo la desventaja es que estos servicios son normalmente de pago.

2.3 Bases de datos

En este apartado vamos a ver las tecnologías MySQL y las bases de datos internas de Android.

2.3.1 MySQL

Sistema de base de datos por excelencia, licenciada bajo GPL/Licencia comercial de Oracle.

Desarrollado en ANCI, C y C++, se trata de un sistema de gestión de base de datos relacional, que incluye varias personalizaciones como: la gestión de usuarios, de tablas, la implementación de vistas en índices, etc. La implementación de estas características hace de Mysql una herramienta potente y a tener en cuenta para su implementación.

2.3.2 MariaDB

Base de datos que se distribuye con la herramienta Xampp, desarrollado por el descontento de algunos desarrolladores con MySQL [20].

La diferencia con MySQL es la posibilidad de realizar consultas más complejas y de forma más rápida, elimina redundancias en la tabla de Chequeos, replicación rápida y segura.

2.3.3 SQLite

Sistema de gestión de base de datos relacional, muy ligera, y escrita en C/C++, que se adapta perfectamente al desarrollo en dispositivos móviles, ya que no necesita un servidor y se integra dentro de la propia aplicación

En Android existen diversas facilidades para la manipulación de este tipo de bases de datos, proporcionando una API completa para la creación, gestión y realización de todas las tareas necesarias.

Otra característica interesante es que la curva de aprendizaje de esta base de datos no es pronunciada, es decir, no es muy difícil aprender a utilizarla.

2.4 Librerías para las conexiones Android con servidores externos

En este apartado vamos a ver las tecnologías necesarias para comunicar nuestra aplicación con servidores externos.

Tenemos que aclarar que al ser nuestra aplicación un servicio REST tendrá que realizar peticiones HTTP a nuestros recursos, y a su vez, tendrá que realizar otras peticiones a los recursos de la página ***www.ultimatecentral.com*** [21] para obtener los datos necesarios para mostrar la información que se pide.

Por otra parte nuestro servidor externo debe ser capaz de proporcionar un servicio web RESTFUL, para, en primer lugar, recoger las peticiones que se envíen desde la aplicación Android; en segundo lugar, procesar la información y realizar las tareas necesarias; y por último, devolver el estado final de la petición.

Por estos motivos se creará una API REST con PHP, MySQl, JSON o XML, para la aplicación, cuya creación y componentes se detallan más adelante.

2.4.1 HttpURLConnection

Esta librería Android, que forma parte del paquete java.net, nos permite recibir y enviar información desde nuestra aplicación hacia un servidor externo, siguiendo el protocolo HTTP [22].

La librería permite a nuestra aplicación actuar como un cliente HTTP ligero, haciendo posible realizar peticiones GET o POST.

Para la petición solo hace falta saber la ubicación del recurso en red, definir qué tipo petición quieres realizar y hacer la llamada. Si el método devuelve algo, el resultado deberá parsearse para poder manipular la información obtenida.

2.4.2 Volley

Librería desarrollada por Google para optimizar la recepción y el envío de peticiones HTTP hacía recursos externos, cuya principal característica es facilitar la tarea del programador cuando realiza este tipo de peticiones, ya que actúa como una interfaz de alto nivel, encargándose de la administración de hilos y procesos tediosos de parsing (Figura 8).

Volley, al igual que la anterior librería, permite a la aplicación actuar como un cliente HTTP para poder comunicarse con el servidor externo, creando tareas asíncronas por cada petición y evitando llamadas y código repetitivo [23]. Entre sus características más importantes se encuentran:

- El procesamiento concurrente, priorización, cancelación y personalización de las peticiones.
- Gestión de tareas en segundo plano, encargándose de la implementación de hilos.
- Implementación de caché en disco y memoria

Sin embargo el uso de Volley se limita cuando hablamos de carga de datos pesados, porque sus operaciones van dirigidas a guardar en caché la información solicitada, por lo que su utilización para este tipos de datos ralentizaría los procesos y la carga.

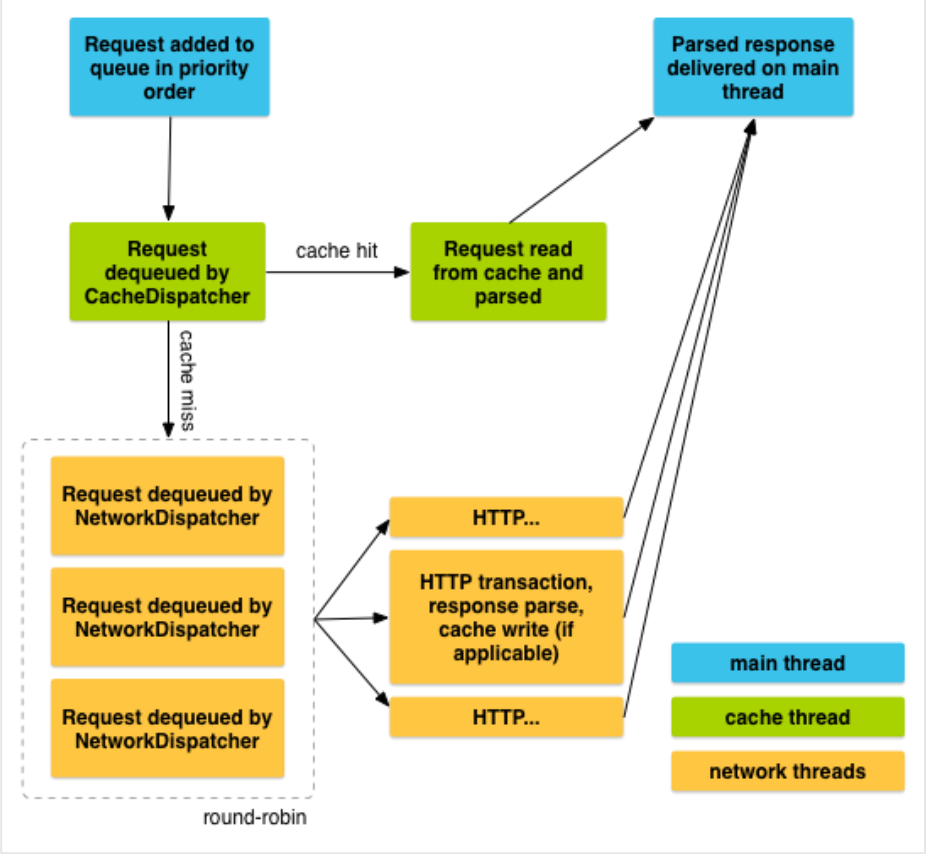


Figura 8: Proceso de una petición Volley

2.4.3 Retrofit

Al igual que las librerías Volley y HttpURLConnection, Retrofit permite la comunicación con servidores externos, ya que convierte a nuestra aplicación en un cliente, de esta forma la comunicación cliente-servidor está asegurada [24].

Esta biblioteca actúa como un cliente REST para las aplicaciones Android, y permite crear todo tipo de conexiones, y realizar todo tipo de peticiones (POST and GET) de forma más simple. Para ello utiliza una clase llamada Retrofit, que contiene el tipo de formato de la respuesta, la URL de la página con la que nos vamos a comunicar, y una interfaz en la que se encuentra la ruta específica del archivo por el que vamos a preguntar. La interfaz servirá para guardar las llamadas que haremos al servidor.

Por último, la librería actúa de forma similar a Volley, ya que se encarga de las tareas más tediosas y de los hilos en segundo plano, creando colas de peticiones para poder manipular las peticiones realizadas.

2.4.4 JSOUP

Librería basada en Java para la realización de Web Scraping de páginas web externas, utilizando para ello las etiquetas y los métodos DOM, CSS y JQuery.

Su función principal es recopilar grandes cantidades de datos diferentes de páginas web a través de código HTML, texto, o archivos, para posteriormente manipularlos y convertirlos al formato que deseamos (parsing).

La librería solo necesita la dirección del archivo, página o texto para empezar el parsing de los datos, utilizando selectores DOM o CSS [25].

2.5 Formato de intercambios de Datos

Este apartado tiene relación con los Apartados 2.2, 2.3 y 2.4, ya que según queramos podemos intercambiar información con varios formatos diferentes: textual, JSON, XML, etc.

En esta sección vamos a ver dos de los principales formatos que se utilizan para el intercambio de datos entre aplicaciones: JSON y XML

2.5.1 JSON

JSON (JavaScript Object Notation*)* es un formato de texto ligero que permite el intercambio de datos entre dos o más aplicaciones, cuya ventaja es la sencillez a la hora

de ser analizado por el analizador sintáctico **¡Error! No se encuentra el origen de la referencia.**[26].

JSON se emplea habitualmente en ambientes donde el flujo de datos entre el cliente y el servidor es muy importante.

Los datos pueden ser:

- Números
- Cadenas de texto
- Tipos booleanos
- Tipos null,
- Vectores
- Objetos

Figura 9: Ejemplo de un archivo JSON

2.5.2 XML

EXtensible Markup Language (XML), o mejor dicho, lenguaje de marcas extensible, es un tipo de formato de texto para representar datos, que permite el intercambio de los mismos entre distintos sistemas [27].

Como característica principal el uso de ‘marcas’ para estructurar la información. De esta forma permite que los datos sean legibles por las diferentes plataformas que intervienen en el proceso, aunque estas no tengan el miso software o hardware instalado.

Las ventajas de esta tecnología son:

- Es extensible, se pueden añadir nuevas marcas u etiquetas una vez finalizado el proceso de diseño y puesta en producción.
- Es sencillo entender su estructura y procesarla.
- Se puede añadir información significativa sobre el tipo de dato que se envía.

- Como hemos dicho anteriormente, permite que dos aplicaciones en entornos distintos puedan comunicarse a pesar no tener el mismo software o hardware instalado.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE Edit_Mensaje SYSTEM "Edit_Mensaje.dtd">

<Edit_Mensaje>
     <Mensaje>
          <Remitente>
               <Nombre>Nombre del remitente</Nombre>
               <Mail> Correo del remitente </Mail>
          </Remitente>
          <Destinatario>
               <Nombre>Nombre del destinatario</Nombre>
               <Mail>Correo del destinatario</Mail>
          </Destinatario>
          <Texto>
               <Asunto>
                    Este es mi documento con una estructura muy sencilla
                    no contiene atributos ni entidades...
               </Asunto>
               <Parrafo>
                    Este es mi documento con una estructura muy sencilla
                    no contiene atributos ni entidades...
               </Parrafo>
          </Texto>
     </Mensaje>
</Edit_Mensaje>
```

Figura 10: Ejemplo de un archivo XML

2.6 Paradigmas de diseño aplicaciones Android

En este apartado abordaremos los paradigmas de diseño que se utilizan actualmente en Android

2.6.1 Google Design

Este paradigma de diseño e iteración está desarrollado por Google y se enfoca a dispositivos que utilicen un sistema operativo Android, aunque el paradigma es extrapolable a otros tipos sistemas, como son las páginas WEB e iOS [28].

La gran característica de Google Material es basar su diseño en objetos materiales, es decir, cada elemento en el diseño se comporta como un objeto real en un espacio y tiempo determinado.

Al tratarse de objetos que intentan imitar la realidad poseen una serie de características propias del mundo real como: profundidades, superficies, sombras, bordes y color. Además, se trata de un diseño que se aproxima a la iteración real de objetos, e intenta ser fiel a las leyes físicas que rigen el mundo natural. Por ello, en este paradigma

todas las animaciones intentan ser lógicas e intentan seguir estas leyes. Por ejemplo, los objetos se pueden superponer pero no se pueden atravesar los unos con los otros.

2.7 Sistemas de diálogo

En la siguiente sección se detallan que son, y para qué sirven los sistemas de diálogo.

2.7.1 ¿Qué es un sistema de diálogo?

Podemos definir un sistema de diálogo hablado (*spoken dialogue systems [29]*) como un sistema informático que recibe como entrada frases en lenguaje natural expresadas de forma oral y que generan como salida frases del lenguaje natural expresadas asimismo de forma oral. La finalidad de estos sistemas es emular el comportamiento inteligente de un ser humano que realiza una tarea concreta.

Se utiliza en la actualidad por diversas empresas para proporcionar información de forma automática, por ejemplo, horarios de salida de aviones, partes meteorológicos, estado de cuentas bancarias [30].

Como podemos apreciar en los sistemas de diálogo hablado se distinguen dos partes muy importantes: el reconocimiento de voz y la síntesis de texto a voz. Estas dos partes fundamentales deben ser capaces de funcionar independientemente la una de la otra, y a la vez deben comunicarse entre sí para dar una respuesta lógica.

En la Figura 11 se detalla los componentes que conforman un sistema de diálogo hablado [3].

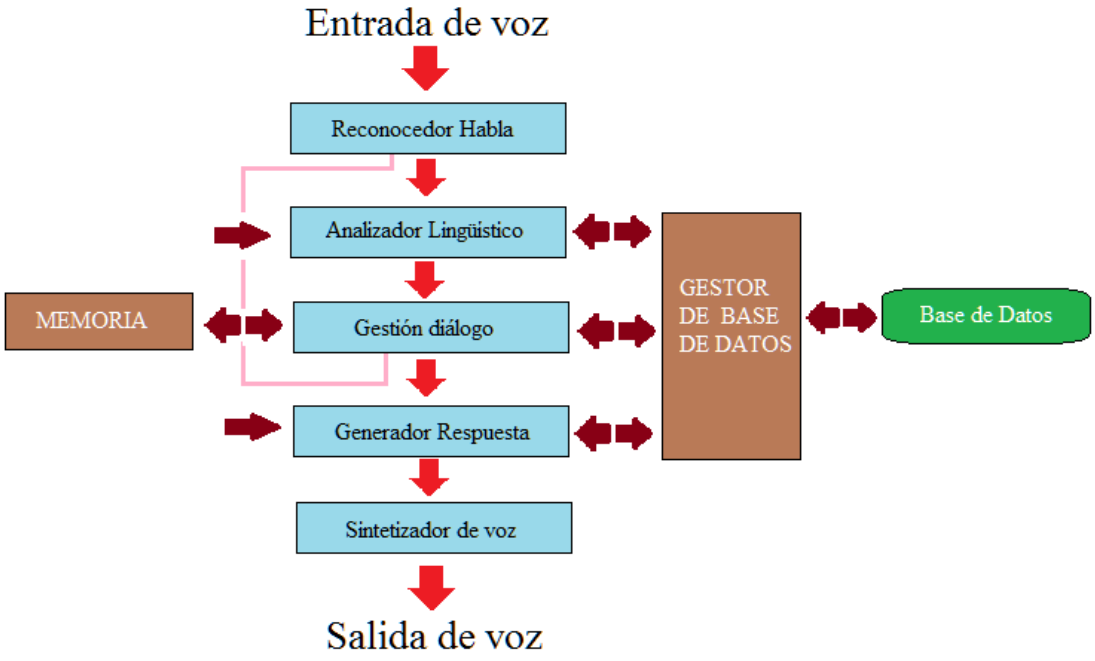


Figura 11: Componentes de un sistema de diálogo

Podemos observar en primer lugar que los módulos que debemos implementar son los siguientes: reconocimiento automático de habla, procesamiento del lenguaje natural, gestión del diálogo, generación de lenguaje natural y síntesis de habla; y estos además pueden comunicarse con otros módulos adicionales propios del sistema como el gestor de la base de datos o la memoria caché, para poder obtener los resultado de las consultas realizadas por el usuario.

Un sistema de diálogo hablado debe seguir el siguiente funcionamiento:

- **Reconocimiento de las elocuciones del usuario**: la entrada del analizador lingüístico son frases o enunciados que el analizador procesará y obtendrá la representación semántica de la frase dicha. El sistema deberá ser capaz de comprender el idioma en el que se está hablando así como las expresiones espontaneas que puedan llegar como entrada.

- **Gestión del diálogo**: la representación semántica de la frase será la entrada para que el gestor del diálogo determine que acción debe realizar el sistema. Este módulo es el encargado de facilitar la interacción entre el usuario y la aplicación, por ello este módulo realizará confirmaciones de datos obtenidos del usuario. Los subdiálogos de corrección y la generación de expectativas respecto a las frases más probables en un momento dado son herramientas que este módulo utiliza para facilitar la interacción y la correcta entrada de datos por parte del usuario.

- **Generador de respuestas:** Este módulo se encargará, una vez definida la acción a realizar, de crear una respuesta en formato de texto. Por su parte el módulo de memoria almacenará las representaciones semánticas que se han obtenido en el transcurso de la interacción, de forma que el analizador lingüístico, el gestor del diálogo y el generador de respuestas puedan acceder al historial. De esta forma podrá realizar: acciones similares cuando el proceso se repita, resolver

referencias anafóricas existentes en las frases enunciadas y puede utilizar información contextual durante la generación de frases.

- **Sintetizador de voz**: Se encargará de recoger la respuesta dada en formato de texto y generar a partir de esta una respuesta oral.

- **Base de datos:** El gestor de base de datos se encargará de realizar las consultas necesarias a la base de datos y recoger el resultado de las mismas, enviando el resultado al módulo de gestión de diálogo.

Los sistemas de diálogo actuales se enfrentan a los siguientes problemas:

- Ser capaces de reconocer el habla espontánea.
- Comprender enunciados sin restricciones de contenido.
- Generar respuestas lógicas, con sentido gramatical, bien formadas y adecuadas al contexto en el que se emite.
- Generar una respuesta totalmente natural y comprensible para el interlocutor.
- Ser multimodal.

Algunos de las aplicaciones de los sistemas de diálogo son las siguientes (Tabla 2)

Contexto	Resumen	Aplicaciones ejemplo
Viajes	Ayuda en la reserva, gestión y cancelación de viajes por medio del habla.	▪ Amtrak ▪ United Airlines ▪ MASK – Multimodal-Multimedia Automated Service Kiosk
Consulta de información	La consulta de información acerca de diferentes índoles.	▪ AdApt – Sistema desarrollado en el Centre for Speech Technology, ▪ August – Sistema desarrollado también en el Centre for Speech Technology, ▪ Júpiter: Sistema de información meteorológica por teléfono. EE.UU.

Aprendizaje	Favorecer el aprendizaje de contenidos docentes por parte de los alumnos.	▪ TRIVIAL - Sistema de diálogo hablado basado en VoiceXML para la enseñanza universitaria
Móviles	Asistentes de voz en las plataformas móviles, para poder navegar por las distintas pantallas.	▪ Asistente de voz Android [31] ▪ SIRI Asistente de voz Iphone [32] ▪ IVONA App de asistencia [33]

Tabla 2: Ejemplos de Sistemas de voz

2.7.2 Interacción oral en Android

Los módulos y funcionalidades descritas para el desarrollo de sistemas de diálogo han evolucionado con Android para poder resolver los problemas que surgen a la hora de implementarlos, por lo que en esta sección haremos un repaso sobre sus componentes y las actualizaciones que estas han sufrido con el tiempo.

Nos encontramos con el primer componente de los sistemas hablado: El reconocimiento de voz.

Reconocimiento de voz

Los dispositivos Android, aunque en sus primeras versiones (*Banana Bread, Cupcake*, *Donut* y *Eclair,* versiones 1.0, 1.5, 1.6 y 2.0 respectivamente) no incorporaban un sistema de reconocimiento de voz, fue a partir de la actualización 2.1 donde se empezó a trabajar con esta tecnología.

En un principio se implementó la búsqueda a través del reconocimiento de voz, e incorporó en el teclado esta funcionalidad añadiendo un botón micrófono, con el que se podía realizar cualquier búsqueda que se quisiera.

En la versión 2.2 Android facilita que otras aplicaciones interaccionen con el reconocimiento de voz, permitiendo que terceras partes proporcionen nuevos motores de reconocimiento, además añade la funcionalidad *Voice Actions for Android* aumentando el número de acciones que puede realizar el usuario con el sistema.

Tendremos que esperar hasta la versión 4.0 para encontrar una actualización significativa, donde se mejora la integración de voz, y el dictado de texto en tiempo real, además de ir mostrando los distintos resultados conforme el usuario expresa la consulta.

Con la actualización 4.1 se mejoran las predicciones de palabras por voz, y el soporte offline para la introducción de información por voz. Por otra parte, se mejora

sustancialmente las acciones que puede realizar los usuarios, además incorpora el asistente de “Google Now”, que nos permite realizar preguntas para luego ser respondidas en un lenguaje natural, gracias a la delegación de las solicitudes a un conjunto de servicios web.

La versión 4.4 también llamada Kit-Kat integra el famoso comando “OK Google” que nos permite acceder al asistente de voz sin tener que acceder al teclado.

En los dispositivos con versiones Android 4.1 o superiores podemos trabajar con el reconocimiento de voz sin necesidad de acceder a Internet, aunque en primer lugar debemos descargar los paquetes de idiomas que vamos a utilizar.

La implementación de la característica de reconocimiento de voz Offline debe ser activada en los dispositivos móviles (Figura 12).

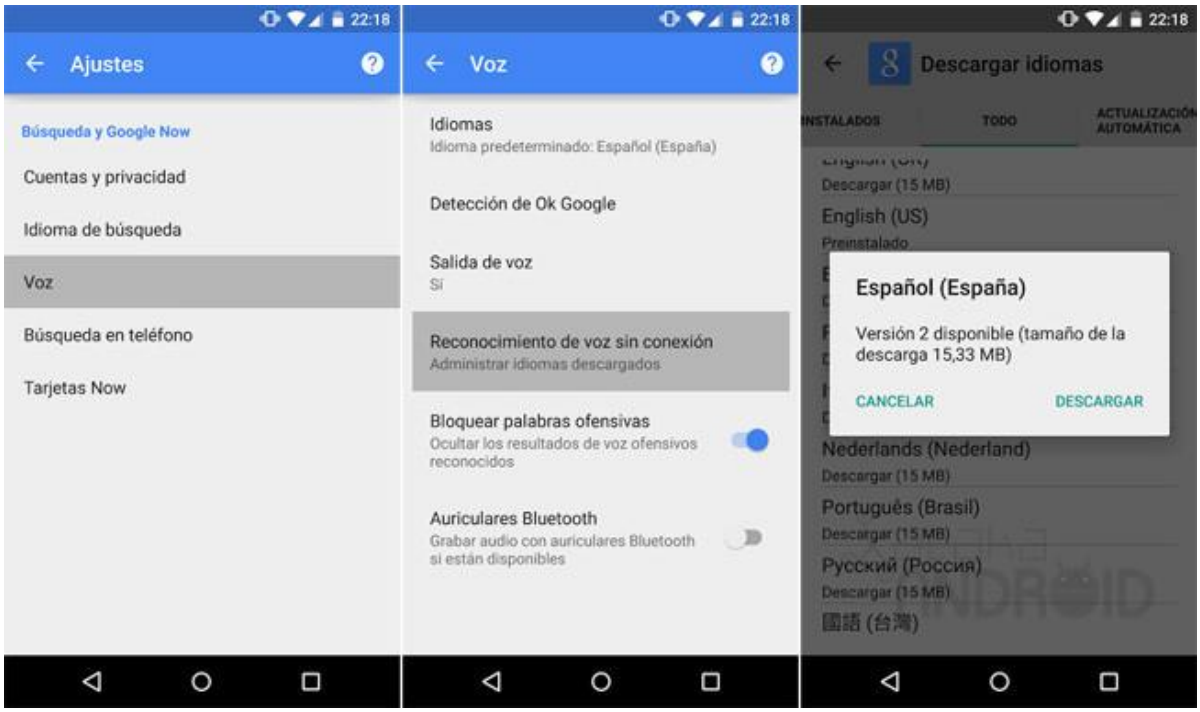


Figura 12: Pasos para activar el reconocimiento de voz en Android

Por último tenemos la última actualización proporcionada por Google, la versión 6.0 o *Marshmallow*.

Marshmallow nos facilita el asistente por voz *Now On Tap*, evolución del Google Now, que nos permite una mejor integración con las aplicaciones. Además, esta versión añade un API que nos permite, una vez una aplicación ha sido lanzada, solicitar una confirmación de voz por parte del usuario, seleccionar una lista de opciones o solicitar las informaciones que necesite.

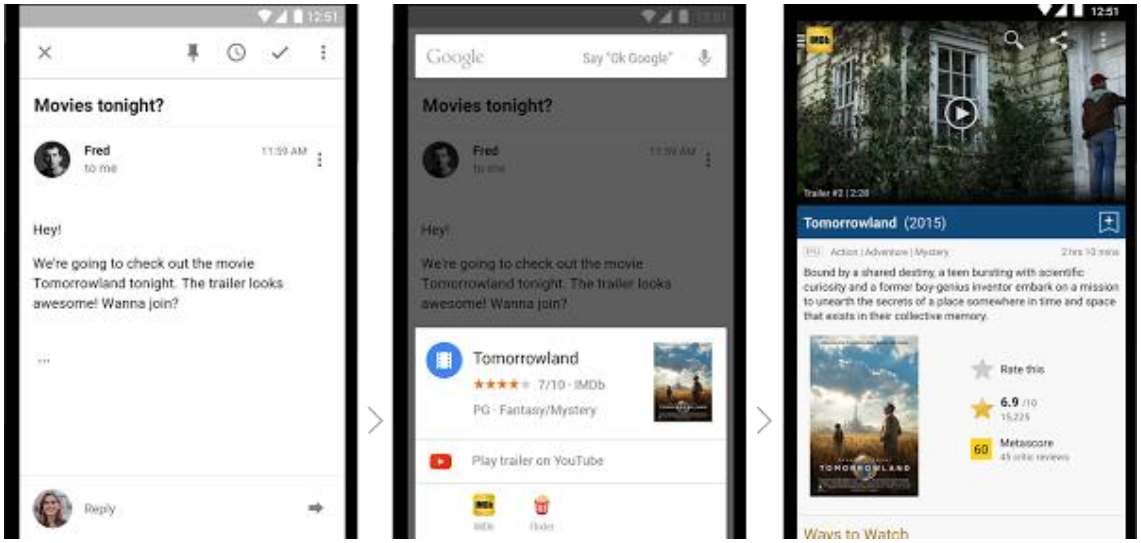


Figura 13: Google On tap Búsqueda de voz

Síntesis de texto a voz

Los sistemas Android incorporan desde sus primeras versiones (a partir de la versión 1.6) un motor de síntesis de voz. Este motor desarrollado por SVOX y Google, permite integrar la síntesis de voz a cualquier aplicación.

Algunos fabricantes que soportan los sistemas operativos Android prefieren utilizar sus propios sintetizadores de voz, llamados también TTS.

Con las distintas actualizaciones el sintetizador de voz de Android (Pico TTS) ofrece un abanico de posibilidades [34].

- Permite que las aplicaciones lean en voz alta el texto que aparece en pantalla.
- Permite instalar archivos de datos de voz para poder reproducir los textos a voz sin necesidad de conexión a internet.
- Permite selecciona la velocidad de voz de reproducción: se puede seleccionar la velocidad a la que se lee el texto.
- Viene con idiomas predefinidos.
- Selección de Idiomas: aunque existe un idioma predeterminado, permite la instalación de otros idiomas para su posterior elección. Pico TTS incorpora los idiomas: Español, Inglés, francés, italiano, alemán, etc.
- La calidad de la voz aunque mejorada con el paso de las actualizaciones puede parecer aún un poco monótona.

Aplicaciones nativas que utilizan este sistema:

- Google Play Libros para utilizar la función Leer en voz alta en tu libro favorito.

- El Traductor de Google para que puedas oír cómo se pronuncia una palabra TalkBack y aplicaciones de accesibilidad para leer en voz alta mensajes en todo el dispositivo.

Para utilizar la síntesis de voz de Google en tu dispositivo con sistema operativo Android, se debe acceder a: a Ajustes > Idioma e introducción de texto > Síntesis de voz, y seleccionar la síntesis de voz de Google como motor preferido (Figura 14).

Hay que tener en cuenta que en muchos dispositivos Android el motor de síntesis de voz de Google ya está disponible, por lo que solamente hace falta en primer lugar activarlo, y luego, descargar o cambiar los paquetes de idiomas por el que se quiera escuchar el asistente de voz.

Por otra parte, el sintetizador de voz de Google nos permite, por medio de los paquetes de descarga, instalar o cambiar diferentes tipos de voces femeninas o masculinas, de esta forma podemos elegir entre 3 voces femeninas y 3 masculinas.

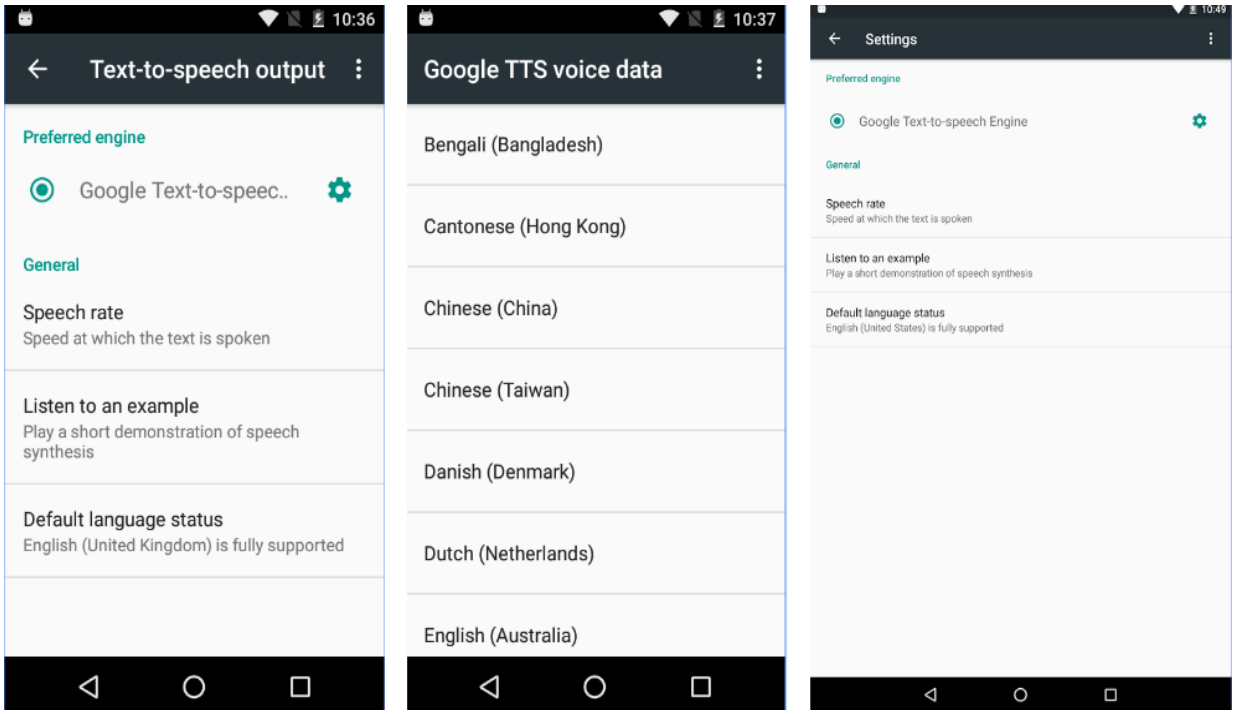


Figura 14: Pasos para activar el asistente de voz en Android

Además del sintetizador por defecto de Android existen otros sintetizadores desarrollados por otras compañías, que se pueden instalar en el sistema operativo. Algunos de los más populares son

- IVONA TTS HQ: Sintetizador de voz muy popular y uno de los mejores que existen en el mercado. Existen versiones de pago y versiones gratuitas del producto. La versión gratuita no implementa algunas características y voces que la opción de pago vienen instaladas. Con IVONA puedes instalar muchos tipos de voz, tanto femenina como masculina, y la calidad de este sintetizador es muy alta.

- Classic Text To Speech Engine: Desarrollado también por SVOX este sintetizador nos permite instalar multitud de voces distintas, de más de 25 lenguajes distintos. Permite entre otras cosas la lectura de mensajes de voz, direcciones de navegación, y la traducción. Es uno de los sintetizadores más populares.

- Talk - Texto a Voz: Permite convertir texto a voz desde cualquier página web en la que se esté navegando, así como de las aplicaciones del dispositivo. Soporta una gran cantidad de lenguajes y tiene la posibilidad de empezar, pausar o para la reproducción que se esté escuchando.

2.7.3 Librería de reconocimiento de voz en Android

Hemos hablado sobre la evolución de los sistemas de diálogo en los dispositivos Android. En este tema nos adentraremos en las tecnologías que nos brinda Android para poder desarrollar la aplicación con reconocimiento de voz.

Las alternativas que existen para desarrollar aplicaciones con reconocimiento de voz son variadas, desde el propio sistema que proporciona una librería nativa para el reconocimiento de voz, hasta APIs externos complementarios o completamente independientes. Algunas de estas librerías son:

Android.Speech: Librería nativa de Android. Es relativamente sencillo incorporar la librería al proyecto. Es una librería muy potente que ofrece diferentes clases para el uso del reconocimiento del habla. Utiliza objetos del tipo RecognizerIntent para poder gestionar el reconocimiento de voz. Estos objetos generan ficheros de audio que serán enviados al servidor de Google para ser procesados, y devolverá el resultado a la actividad de la que se lanzó el RecognizerIntent.

AT&T Speech API: Es un API desarrollado por la empresa AT&T, y consiste en una librería que nos da un soporte online para el reconocimiento de voz. Es decir, es necesario estar conectado a la red para poder realizar el proceso del reconocimiento de voz. Es una librería potente diseñada para varios tipos de sistemas operativos, y es de pago.

CMU Sphinx: es un conjunto de librerías desarrolladas para el reconocimiento de voz, que nos permite integrarla en dispositivos con sistema operativos Android. Ha sido desarrollado por la Sphinx Group del *Carnegie* Mellon University. Soporte múltiples lenguajes además de poder trabajar de forma offline.

Nuestros esfuerzos se van a centrar en analizar la librería Android.speech, ya que de todas las alternativas es la que más se adapta las necesidades de la aplicación, además de estar muy integrado con el sistema operativo Android y tener una documentación muy detallada.

Android.Speech

Android.Speech proporciona una serie de clases e interfaces para la integración de la librería al proyecto. (Tabla 3 y 4); solo hace falta la importación de la librerías a los módulos o clases que van a hacer las llamadas [35].

Interfaces	
RecognitionListener	Usado para recibir las notificaciones de la clase *SpeechRecognizer* cuando un evento que es reconocido por este último ocurre.

Tabla 3: Interfaces de la librería Android.Speech

Clases	
RecognitionService	Esta clase provee una clase base para la implantación de servicios de reconocimiento. La clase debe utilizar solo en el caso de que se quiera implementar un nuevo reconocedor del habla.

Clases	
RecognitionService.Callback	Esta clase recibe las devoluciones de llamadas del servicio de reconocimiento del habla y las envía al usuario.
RecognizerIntent	Constantes necesarias para apoyar el reconocimiento de voz, cuando se inicia desde un *intent.*
RecognizerResultsIntent	Define las contantes necesarias para aquellos *intens* que se encargarán de recoger y mostrar los resultados del reconocimiento de voz. Sólo deberán ser llamados si se desea una vista de los resultados del reconocimiento de voz, o si se desea ofrecer una visión diferente de los resultados; por lo que no debería ser llamado para el uso normal del reconocimiento de voz.
SpeechRecognizer	Esta clase provee el acceso al servicio de reconocimiento de voz. El servicio permite el acceso al reconocedor de voz (speech recognizer). Esta clase no debe ser instanciada directamente, se debe llamar al método *createSpeechRecognizer(Context).* Esta clase debe ser llamada desde hilo principal. Además esta API no está enfocada para el uso continuo del reconocimiento de voz, ya que consume una cantidad considerable de memoria.

Tabla 4: Clases de la librería Android.Speech

De las clases listadas anteriormente nos centraremos en la clase que hace posible el implementar el reconocimiento de voz en nuestra aplicación: *RecognizerIntent.*

Para poder utilizar esta clase debemos importar a nuestras actividades la siguiente línea de código: *import android.speech.RecognizerIntent*.

Esta importación hará posible que nuestro dispositivo pueda utilizar el reconocimiento de voz. Para ello debemos crear un *intent* con el parámetro *RecognizerIntent.ACTION_RECOGNIZE_SPEECH* (API level: 24) para poder iniciar la actividad de reconocimiento de voz por parte del usuario.

La tabla 5 explica detalladamente las distintas constantes que puede utilizar la clase *RecognizerIntent.*

Constante	Tipo	Función
ACTION_GET_LANGUAGE_DETAILS	String	*intent* de tipo broadcast que envía un objeto BroadcastReceiver para solicitar la lista de los idiomas disponibles.
ACTION_RECOGNIZE_SPEECH	String	Empieza una actividad para empezar el reconocimiento de voz por parte del usuario a través del *speech reconigzer*.
ACTION_VOICE_SEARCH_HANDS_FREE	String	Empieza una actividad para empezar el reconocimiento de voz si un requerimiento visual o un entrada de "touch"
ACTION_WEB_SEARCH	String	Empieza una actividad para empezar el reconocimiento de voz por parte del usuario y realiza una búsqueda web y lo muestra.
DETAILS_META_DATA	String	Nombre de metadatos bajo las cuales una actividad *ACTION_WEB_SEARCH* puede utilizar para revelar el nombre de la clase *BroadcastReceiver*.
EXTRA_CALLING_PACKAGE	String	Clave extra que se usa en un *intent* para el uso del reconocimiento de voz por parte del *speech recognizer*.
EXTRA_CONFIDENCE_SCORES	String	Array que contiene valores de tipo float que representan la fiabilidad de cada resultado devuelto por el intent ACTION_RECOGNIZE_SPEECH.
EXTRA_LANGUAGE	String	Lenguaje opcional de tipo IETF
EXTRA_LANGUAGE_MODEL	String	Informa al reconocedor cual es la preferencia actual del tipo de modelo de habla cuando al realizarse la acción ACTION_RECOGNIZE_SPEECH.
EXTRA_LANGUAGE_PREFERENCE	String	Clave extra que identifica el idioma preferente del usuario.
EXTRA_MAX_RESULTS	String	Límite opcional del límite máximo de resultados a devolver.
EXTRA_ONLY_RETURN_LANGUAGE_PREFERENCE	String	Valor booleano que se utiliza como suplemento adicional al *intent* ACTION_GET_LANGUAJE_DETAILS cuando el único resultado devuelto es el EXTRA_LANGUAJE_PREFERENCE.
EXTRA_ORIGIN	String	Valor opcional el cual es usado para indicar la URL de la página de la que se solicitó el habla.
EXTRA_PARTIAL_RESULTS	String	Valor booleano opcional que indica si se debe devolver los resultados mientras el usuario realiza la consulta hablada.
EXTRA_PREFER_OFFLINE	String	Valor booleano que indica si se debe utilizar un único motor de reconocimiento en línea.
EXTRA_PROMPT	String	Texto opcional que se muestra al usuario cuando se le pide que hable.
EXTRA_RESULTS	String	Un *ArrayList* de tipo *String* que contiene los resultados de la acción ACTION_RECOGNIZE_SPEECH.
EXTRA_RESULTS_PENDINGINTENT	String	Cuando la acción realizada es ACTION_RECOGNIZE_SPEECH, la actividad que contiene la entrada del habla

Constante	Tipo	Función
		devolverá los resultados a través de los mecanismos de resultados de la actividad.
EXTRA_RESULTS_PENDINGINTENT_BUNDLE	String	Si se utiliza EXTRA_RESULTS_PENDINGINTENT para suministrar un intento de reenvío, también puede utilizar este accesorio para suministrar suplementos adicionales para obtener el propósito final.
EXTRA_SECURE	String	Valor booleano opcional que indica que una búsqueda por voz *hands free* se llevó a cabo mientras el dispositivo estaba en un modo seguro.
EXTRA_SPEECH_INPUT_COMPLETE_SILENCE_LENGTH_MILLIS	String	Tiempo total que se debe tomar después de cesar la captura de voz para considerar la entrada completa.
EXTRA_SPEECH_INPUT_MINIMUM_LENGTH_MILLIS	String	La longitud mínima de un enunciado.
EXTRA_SPEECH_INPUT_POSSIBLY_COMPLETE_SILENCE_LENGTH_MILLIS	String	Tiempo total que se debe tomar después de cesar la captura de voz para considerar la entrada posiblemente completa.
EXTRA_SUPPORTED_LANGUAGES	String	Clave que identifica el contenido extra del resultado devuelto por el intent ACTION_GET_LANGUAJE_DETAILS a través del BroadcastReceiver.
EXTRA_WEB_SEARCH_ONLY	String	Valor booleano opcional, que se utiliza con ACTION_WEB_SEARCH para indicar si se debe sólo mostrar los resultados de la búsqueda en la web.
LANGUAGE_MODEL_FREE_FORM	String	Establece el lenguaje basado en *FREE FORM* para el dictado.
LANGUAGE_MODEL_WEB_SEARCH	String	Utilizar un modelo de lenguaje basado en los términos de búsqueda web.
RESULT_AUDIO_ERROR	Int	Código resultado devuelto cuando se ha detectado un error de audio
RESULT_CLIENT_ERROR	Int	Código resultado devuelto cuando hay un error genérico por parte del cliente
RESULT_NETWORK_ERROR	Int	Código resultado devuelto cuando se ha encontrado un error en la red
RESULT_NO_MATCH	Int	Código resultado devuelto cuando no se encuentran coincidencias para el discurso dictado.
RESULT_SERVER_ERROR	int	Código resultado devuelto cuando el servidor de reconocimiento de voz devuelve un error

Tabla 5: Clases RecognizerIntent

La integración de Android.Speech al ser una librería que se incluye con la distribución de Android Studio, consiste únicamente en importarla a las actividades donde vamos a realizar el reconocimiento de voz.

En primer lugar Android.Speech es una librería que necesita de la conexión a internet para su utilización (si no tenemos descargado el paquete de reconocimiento de un idioma

específico), por lo que la primera acción a realizar es dar los permisos necesarios a la aplicación para poder conectarse a Internet:

<uses-permission android:name="android.permission.INTERNET" />

Una vez tenemos el permiso necesario podemos declarar nuestros *intens* y poder gestionar la entradas por audio. Esta interacción se recoge de esta forma:

- En primer lugar importamos la clase que vamos a utilizar:

  import android.speech.RecognizerIntent

- Creamos un botón micrófono para iniciar la entrada por voz.

  ```
  < ImageButton
     android:id="@+id/action_dar_instrucciones"
     android:icon="@drawable/microphone"
     android:title="Micrófono"
  app:showAsAction="always"/>
  ```

- Capturamos el botón de escucha:

  ```
  reconoce_habla = (ImageButton) view.findViewById(R.id.
  action_dar_instrucciones);
  ```

- Capturamos el inicio de la actividad, y creamos los intens necesarios.

  ```
    @Override
  public void onClick(View v) {
  ….
  case R.id.img_btn_hablar_login:

      Intent intent = new Intent
      (RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
  ```

- Añadimos las opciones extras que queramos, en este caso el idioma por el cuál vamos a reconocer: el español.

  intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL, "es");

O un idioma en forma libre:

```
intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,
RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);
```

- Iniciamos la actividad que recogerá la entrada por voz del usuario. Si el dispositivo soporta la entrada por voz, la actividad se iniciará; sin embargo, si el dispositivo no soporta entradas por voz devolverá un error y aparecerá un mensaje indicando esta situación.

```
try {
  startActivityForResult(intent, RECOGNIZE_SPEECH_ACTIVITY);
}
catch (ActivityNotFoundException a) {
  Toast.makeText(getActivity().getApplicationContext(), "Tu dispositivo no soporta el reconocimiento de voz", Toast.LENGTH_LONG).show();
 }
```

- Una vez procesado la información por el reconocimiento de voz de Google, este devolverá un resultado a la actividad inicial de donde partió dicha información. Debemos recoger y tratar el resultado mediante un método, con el que podemos comprobar si el resultado es correcto o no lo es. Este método es: onActivityResult que recibe como parámetros el resultado de la consulta (requestcode).

```
public void onActivityResult(int requestcode, int resultcode, Intent datos)
  {
    if (resultcode== Activity.RESULT_OK && datos!=null)
        {

       ArrayList<String>
text=datos.getStringArrayListExtra(RecognizerIntent.EXTRA_RESULTS
);

///Procesamiento de la información obtenida.

        }

  }
```

- Si los resultados son correctos los datos devueltos están contenidos en un array, por lo que debemos procesarlos de la forma que mejor nos convenga para obtener los resultados queridos.

Debemos aclarar que las actividades deben ser capaces de escuchar las acciones del micrófono, por este motivo estas clases deben estar implementadas por la clase *View.OnClickListener*.

```
public class LoginFragment extends Fragment implements View.OnClickListener{

……}
```

2.7.4 Librería de síntesis de voz en Android

Vamos a ver las posibilidades de integrar un motor de síntesis de voz en nuestro proyecto Android.

Android ofrece, como con el reconocimiento de voz, una librería nativa para poder utilizar este tipo de tecnologías en nuestras aplicaciones: ***android.speech.tts***

Esta librería nos permite entre sus funcionalidad introducir una cadena de texto, por medio de su clase ***android.speech.tts.TextToSpeech,*** para su reproducción por medio de los altavoces del dispositivo.

Esta librería utiliza colas de reproducción para la reproducción de los textos, de esta forma dos audios no están superpuestos el uno al otro.

Las interfaces y clases que conforman esta librería son las siguientes:

Interfaces	
SynthesisCallback	Llamada para retornar los datos de la voz sintetizada por un motor de síntesis de voz.
TextToSpeech.OnInitListener	Interfaz de definición de una llamada para indicar la invocación de la inicialización de una conversión de texto a audio.

Tabla 6: Interfaces de la librería android.speech.tts

Clases	
SynthesisRequest	Contiene los datos requeridos por los motores de síntesis de voz.
TextToSpeech	Sintetiza la voz a partir de un texto para su reproducción inmediata, o crea un archivo de audio.
TextToSpeech.Engine	Constantes y nombres de los parámetros para el control de texto a voz.
TextToSpeech.EngineInfo	Información acerca del motor de conversión de texto a voz instalado
TextToSpeechService	Clase base abstracta para implementación de un motor TTS.
UtteranceProgressListener	Clase para capturar los eventos relacionados con el progreso de un enunciado a través de la cola de síntesis.
Voice	Las características y funciones de una voz *Text-To-Speech*

Tabla 7: Clases de la librería android.speech.tts

La integración de Android.Speech.tts, al ser una librería que se incluye con la distribución de Android Studio, consiste en importar la librería en las actividades donde vamos a realizar la síntesis de voz. De forma más concreta debemos importar la clase Android.Speech.tts.TextToSpeech, ya que será esta clase la que nos permite invocar los métodos necesarios para la inicialización e introducción de textos en la cola de reproducción.

Al igual que Android.Speech, esta librería no necesita permisos especiales para su utilización. Solo debemos realizar los siguientes pasos para poder utilizarla

- En primer lugar importamos la clase que vamos a utilizar:

```
import android.speech.tts.TextToSpeech;
```

- Creamos un botón sonido para la reproducción de la voz.

```
< ImageButton
      android:id="@+id/action_oir_informacion"
      android:icon="@drawable/voice"
      android:title="Sonido"
 app:showAsAction="always"/>
```

- Capturamos el botón de sonido:

```
reconoce_habla = escucha_info= (ImageButton) view.findViewById(R.id.
action_oir_informacion);
```

- Capturamos el inicio de la actividad mediante un *listener*.

```
TextToSpeech.OnInitListener listener = new
TextToSpeech.OnInitListener(){
   @Override
   public void onInit(int status) {….}
```

- Cuando inicializamos comprobamos primero el estatus, ya que la clase debe devolver si se ha realizado la conversión de forma correcta si no es así entonces ha fallado.

```
if (status == TextToSpeech.SUCCESS){}
else {Log.e("TTS", "Initilization Failed!");
```

- Una vez que hemos comprobado que no falla pasamos a pasar los parámetros que van a configurar nuestro tts como por ejemplo es tipo de lenguaje:

```
Locale spanish = new Locale("es", "ES");
int result = tts.setLanguage(spanish);
if (result == TextToSpeech.LANG_MISSING_DATA || result ==
TextToSpeech.LANG_NOT_SUPPORTED) {
        Log.e("TTS", "This Language is not supported");
}
if (Build.VERSION.SDK_INT >=
Build.VERSION_CODES.LOLLIPOP) {
     String bienvenida =
     tts.speak(bienvenida , TextToSpeech.QUEUE_FLUSH, null, null);
  }else{
     String bienvenida =
```

```
            getResources().getString(R.string.MensajeBienvenida_login);
            tts.speak(bienvenida, TextToSpeech.QUEUE_FLUSH, null);
        }
```

- Podemos observar que si el lenguaje no es soportado dará un error, además si la versión del dispositivo es menor que *Lollipop* (5.0) invocara otro método para la reproducción de texto.

- Para terminar la inicialización pasamos el *listener* a la variable TTS que hemos creado antes, además de la actividad donde nos encontramos para que la clase sepa cuál es la actividad desde donde se invoca la escucha.

```
tts = new TextToSpeech(getActivity(), listener);
```

- Esta primera parte se refiere a la inicialización y al saludo de bienvenida una vez activada la clase. Sin embargo podemos crear un método en la Actividad principal para llamar al TTS cada vez que necesitemos reproducir un texto. Este método se llamara *speak*, y utilizará el método *speak* del TTS principal, para añadir el texto que viene como parámetro a la cola de reproducción.

```
    private void speak(String text){
        if (Build.VERSION.SDK_INT >=
         Build.VERSION_CODES.LOLLIPOP) {
            tts.speak(text, TextToSpeech.QUEUE_FLUSH, null, null);
        }else{
            tts.speak(text, TextToSpeech.QUEUE_FLUSH, null);
        }
    }
```

- QUEUE_FLUSH es un modo que permite que todas las entradas de reproducción se eliminen y se sustituyan por una nueva entrada.

- Algunas veces tendremos que forzar el paro de las reproducciones, ya sea porque el usuario ha decidido parar manualmente el audio o debido a un cambio de actividad (Ejemplo ir de Registro a Ver torneos), para ello contamos con el método: *tts.stop();* que nos permite parar la reproducción del audio en cualquier momento.

2.8 Realidad Aumentada

En esta sección veremos en qué punto se encuentran y que tipo de tecnologías integra la Realidad Aumentada.

2.8.1 ¿Qué es la Realidad Aumentada?

La Realidad Aumentada (RA) o *Augmented Reality* (AR) [36] es el término que se utiliza a para definir una visión a través de un dispositivo tecnológico, directa o indirecta, de un entorno físico del mundo real, cuyos elementos se combinan con elementos virtuales para la creación de una realidad mixta en tiempo real.

La Realidad Aumentada nos acerca las informaciones de puntos específicos de la realidad y la mezcla con la misma gracias a los dispositivos que se encuentran en los aparatos electrónicos, tales como cámaras de video, fotos, gps, etc.

Las características que debe tener una aplicación de realidad aumentada son las siguientes (en palabras de Ronald Azuma) [37]:

- Combinación de elementos reales y virtuales.
- Es interactiva en tiempo real.
- Está registrada en 3D.

Otra definición sobre qué servicios debe implementar una aplicación AR la da la empresa telefónica [5]. Los elementos o características son las siguientes:

- Por un lado, un elemento que capture las imágenes de la realidad que están viendo los usuarios. Basta para ello una sencilla cámara de las que están presentes en los ordenadores o en los teléfonos móviles.

- Por otro, un elemento sobre el que proyectar la mezcla de las imágenes reales con las imágenes sintetizadas. Para ello se puede utilizar la pantalla de un ordenador, de un teléfono móvil o de una consola de videojuegos.

- En tercer lugar, es preciso tener un elemento de procesamiento, o varios de ellos que trabajan conjuntamente. Su cometido es el de interpretar la información del mundo real que recibe el usuario, generar la información virtual que cada servicio concreto necesite y mezclarla de forma adecuada. Nuevamente encontramos en los PCs, móviles o consolas estos elementos.

- Finalmente se necesita un elemento al que podríamos denominar «activador de realidad aumentada». En un mundo ideal el activador sería la imagen que están visualizando los usuarios, ya que a partir de ella el sistema debería reaccionar. Pero, dada la complejidad técnica que este proceso requiere, en la actualidad se utilizan otros elementos que los sustituyen. Se trata entonces de elementos de localización como los GPS que en la actualidad van integrados en gran parte de los Smartphone, así como las brújulas y acelerómetros que permiten identificar la posición y orientación de dichos dispositivos, así como las etiquetas o marcadores del tipo RFID o códigos bidimensionales, o en general cualquier otro elemento que sea capaz de suministrar una información equivalente a la que proporcionaría lo que ve el usuario, como por ejemplo sensores.

En un caso ideal, algunos de estos elementos podrían llegar a eliminarse. Esto ocurriría si se consigue, por ejemplo, proyectar la información sintetizada de forma que el ojo sea capaz de verla, bien sobre unas gafas, directamente sobre la retina, o con alguna

técnica holográfica avanzada. Pero, por el momento, esto deberíamos considerarlo todavía futurista.

Los teléfonos con sistema operativo Android cumplen todos requisitos para poder desarrollar aplicaciones de realidad aumentada, aunque estos dispositivos tienen requerimientos mínimos y limitaciones para implantar esta tecnología:

- El uso de la RA está orientado a dispositivos avanzados
- Algunas veces los datos de localización son imprecisos e inexactos.
- Está limitados al contexto para el que se utilice
- Problemas de privacidad.

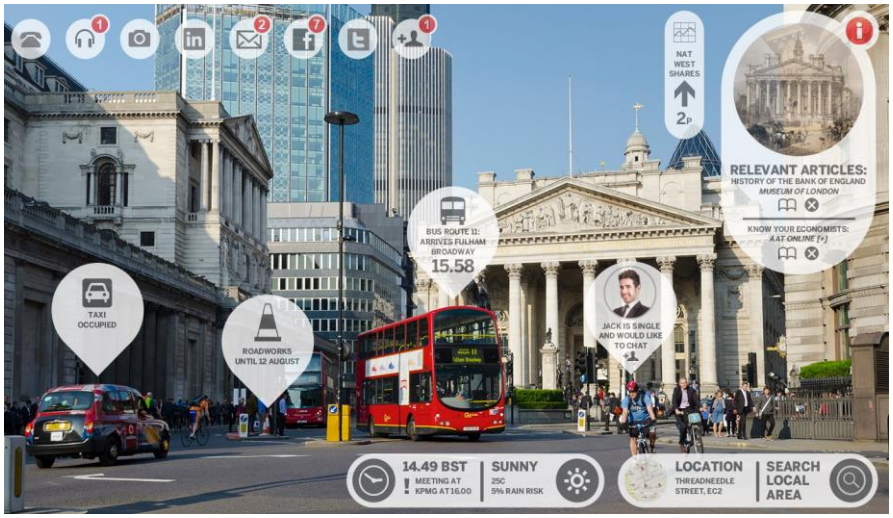


Figura 15: Ejemplo de una interfaz de Realidad Aumentada

Algunos de los ámbitos para los que se utiliza esta tecnología son:

Ámbito	
Videojuegos	PokemónGO: aplicación para buscar “pokemons” por la ciudad. Pacman: juego que consiste en ir comiendo bolas como recompensa.
Enseñanza	Cálculo de volúmenes. Identificación de parte del cuerpo
Marketing y venta	Compras online. TiSSoT.
Viajes y guías turísticas	TripAdvisor: aplicación para ver los restaurantes más cercanos. WIKITUDE Travel guide: guía turística.
Procesos de mantenimiento	Reparaciones de coche. Reparaciones de ascensores.
Procesos de búsqueda	Wikipedia en la aplicación de realidad aumentada *layar*.

Tabla 8: Ejemplos de Realidad Aumentada en aplicaciones

2.8.2 Tecnologías para el desarrollo RA en Android

En Android existen diversas plataformas para el desarrollo de tecnologías de realidad aumentada:

WIKITUDE

Es una tecnología de Realidad Aumentada dirigida a dispositivos móviles, creada en 2008. Inicialmente su meta era proveer una experiencia de realidad aumentada basada en la geolocalización. Para ello utilizaba un navegador por el cuál podías conectarte a la realidad y ver información relevante de la misma.

En la actualidad esta herramienta proporciona su propio SDK que permite una experiencia de realidad aumentada por medio de la geolocalización, así como el reconocimiento de objetos en la realidad, y herramientas para el desarrollo de imágenes 3D para complementar la experiencia.

El principal atractivo del SDK es que es accesible para el desarrollo de RA desde diversas plataformas: JavaScript (con HTML, CSS y JQuery) y nativas (Android e iOS). También provee una serie de herramientas online como complemento a los reconocimientos de imagen y geolocalización, aunque son limitados en versión libre.

Existe una versión de pago y libre para desarrolladores. La versión libre tiene un número limitado de imágenes que puedes reconocer, y con respecto a la geolocalización, en la cámara aparece siempre una huella de agua con el nombre: *trail*.

Figura 16: Ejemplo de Geolocalización y Reconocimiento de Imagen en Wikitude

LAYAR

Fundada en 2009 layar se caracteriza, al igual que WIKITUDE, en ser una herramienta para la utilización de realidad aumentada por medio de la geolocalización.

Para ello utiliza un buscador que se conecta a un servidor por medio de servicios REST para obtener los puntos de interés cercanos, y así dibujarlos en la pantalla.

Además permite el escaneo de imágenes siempre y cuando tengan el Logo de Layar, y permite el escaneo de código QR a través de la cámara del dispositivo.

Es de pago, con 30 días en versión gratuita para los desarrolladores que empiezan.

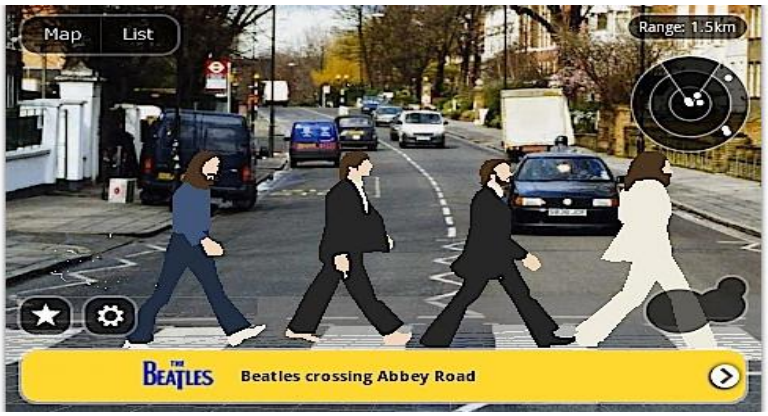


Figura 17: Ejemplo de una aplicación en Layar

VUFORIA

Tecnología de realidad dirigida especialmente al reconocimiento de imágenes, formas y objetos a través de la cámara del dispositivo.

Permite superponer objetos en 3D sobre la interfaz de la cámara, de esta forma, dependiendo de qué objeto se trate, Vuforia lo reconoce y superpone una imagen en 3D que tiene relación con la imagen. Además permite la realización de otras series de acciones cuando la imagen es reconocida.

Posee un SDK para el desarrollo de RA, con herramientas online para el soporte de esta tecnología, y al igual que WIKITUDE existe una versión de pago y libre para desarrolladores. Con la versión libre tenemos limitado el reconocimiento de imágenes.

Figura 18: Ejemplo de superposición en Vuforia

ARTOOLKIT

Es una librería *open-Source* creada en 1999 para la creación de aplicaciones en realidad aumentada que superpone imágenes virtuales en mundo real a través de la cámara del dispositivo, aunque no implementa la función de geolocalización.

Puede reconocer patrones y superponer objetos en 2D y 3D cuando es reconocida la imagen.

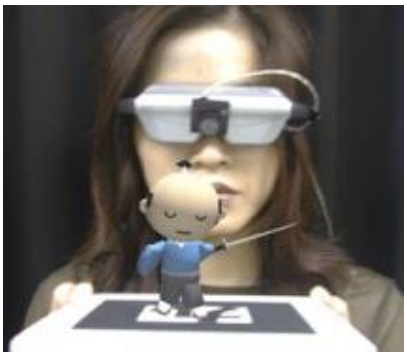


Figura 19: Ejemplo de superposición en ARTOOLKIT

2.8.3 Wikitude. Especificaciones técnicas e integración

Vamos a ver que especificaciones mínimas deben tener un dispositivo para la integración de la librería Wikitude en Android:

Realidad Aumentada	Características necesarias
Geolocalización	▪ Android 4.0.3+ (API Level 15+) ▪ Compass ▪ GPS and / or network positioning ▪ Accelerometer ▪ High resolution devices (hdpi) ▪ Rear-facing camera ▪ OpenGL 2.0

Tabla 9: requisitos Android Wikitude

Además el móvil debes contar al menos de:

- 2GH de procesador.
- 1GB de memoría RAM.
- GPS Alta precisión.
- 3G o 4G para conexiones sin WIFI.
- Cámara de alta resolución.
- Pantalla táctil.

Integración de Wikitude en un proyecto Android.

Wikitude provee a los desarrolladores clases y ejemplos que sirven como modelos para la creación de aplicaciones propias [38].

En nuestro caso el modelo que necesitamos es el modelo base de creación de un entorno de Realidad Aumentada, que interactúa con actividades nativas de la aplicación. Para ello contamos con las siguientes clases:

- *AbstractArchitectCamActivity.java;*
- *ArchitectViewHolderInterface.java;*
- *LocationProvider.java*
- *nativeDetails.js*
- *index.html*
- *ade.js*

Podemos observar que existen clases nativas (Java) y clases relacionadas a las tecnologías web. Esto se debe a que Wikitude se apoya en estas tecnologías para facilitar la integración de aplicaciones AR en distintos dispositivos.

Las aplicaciones RA de Wikitude, también llamadas **World Architect**, se basan en páginas HTML que se integran con la cámara del dispositivo y utilizan JavaScript para llamar a los métodos cuando se selecciona un elemento de la RA.

Clases Nativas

La generación del entorno RA se basa en la creación del componente ArquitectView en Java. Este componente instancia los componentes gráficos (cámara y motor gráfico) y el motor web, para que sirva como punto de entrada de la realidad aumentada. Se encarga además de iniciar, recoger y controlar los eventos RA; por ello la clase principal de nuestra aplicación extenderá de esta clase y por consiguiente heredará los métodos que hacen posible la RA.

La clase *AbstractArchitectCamActivity.java* será la clase que contenga los métodos para generar la RA, por lo que en su interior debe encontrarse la clase ArquitectView de Wikitude. Por otra parte *AbstractArchitectCamActivity.java* debe tener en su interior la clave que proporciona Wikitude a los desarrolladores para poder inicializar el motor AR.

Vamos a pasar a describir brevemente las clases:

- ArchitectViewHolderInterface.java

Interfaz de los métodos que vamos a utilizar para llamar a la actividad de tipo *AbstractArchitectCamActivity.java*.

- LocationProvider.java

Implementación de la clase que nos ayudará a obtener la información de donde nos encontramos, es decir, se comunicará con el GPS para poder obtener la ubicación actual

y pasárselo a la actividad principal. Este archivo se puede modificarse para ajustar los parámetros y los métodos a las necesidades del programador.

- AbstractArchitectCamActivity.java

Clase Abstracta de la que nuestra Actividad principal heredara los métodos y constantes para poder comunicarse con los archivos HTML y JS que crearán el entorno de realidad virtual por medio de la cámara del dispositivo. Además instancia la clase *ArquitectView* para la creación de la cámara.

Esta clase también se puede modificar para adaptar las necesidades del proyecto, pero los métodos que deben existir para iniciar y controlar los eventos de la RA deben ser (Figura 20):

- OnCreate: para inicializar la actividad. Se instancian la posición inicial (LocatioProvider), la cámara y la interfaz web.
- onPostCreate: Se encarga de cargar el entorno de realidad aumentada, además de arrancar el servicio web, y establece los servicios de comunicación entre JS y Java.
- onDestroy: Libera todos los recursos utilizados por la realidad aumentada
- OnResume: Este método recibe todas la modificaciones del entorno una vez la actividad se ha iniciado. Por ejemplo un cambio en la posición del usuario.

JavaScript y HTML

Como se ha mencionado anteriormente, Wikitude obliga la utilización de elementos JavaScript y HTML para poder comunicarse con la librería y mostrar el entorno RA.

Estos elementos permiten capturar los eventos sucedidos a través de la cámara del dispositivo y la geolocalización del usuario. Estos se capturan a través de métodos JavaScript y son comunicados a la aplicación, la cual realizará las acciones pertinentes: como visualizar los puntos de interés o ver la distancia entre dos puntos.

Por último los archivos de esta tecnología se encuentran separados del resto de componentes Java.

- nativeDetails.js

Archivo de tipo JavaScript que invoca *index.html* y que sirve para crear los marcadores y localizaciones que tenemos en nuestra bases de datos. Estos marcadores serán los puntos de interés (POI) que se mostrarán en la pantalla de la cámara y con los que podemos interactuar para obtener su información.

Este archivo debe comunicarse con la base de datos de nuestro servidor local para obtener los datos para dibujar la información en la pantalla del dispositivo, por lo que contiene la dirección URL de nuestro API que devuelve dicha información.

Esta clase se encargará además de enviar a nuestra actividad nativa (principal) los parámetros para que esta última realice las acciones que debe realizar.

- Index.html

Es la página donde se cargan la actividad nativeDetails.js y ade.js, además de las hojas de estilo que vamos a utilizar para representar la interfaz web. En el *index* podemos crear las los elementos y *divs* necesarios para la visualización de los datos.

- ade.js

Librería wikitude que nos permite la simulación del entorno web, y por ende la creación de los componentes que este conforma.

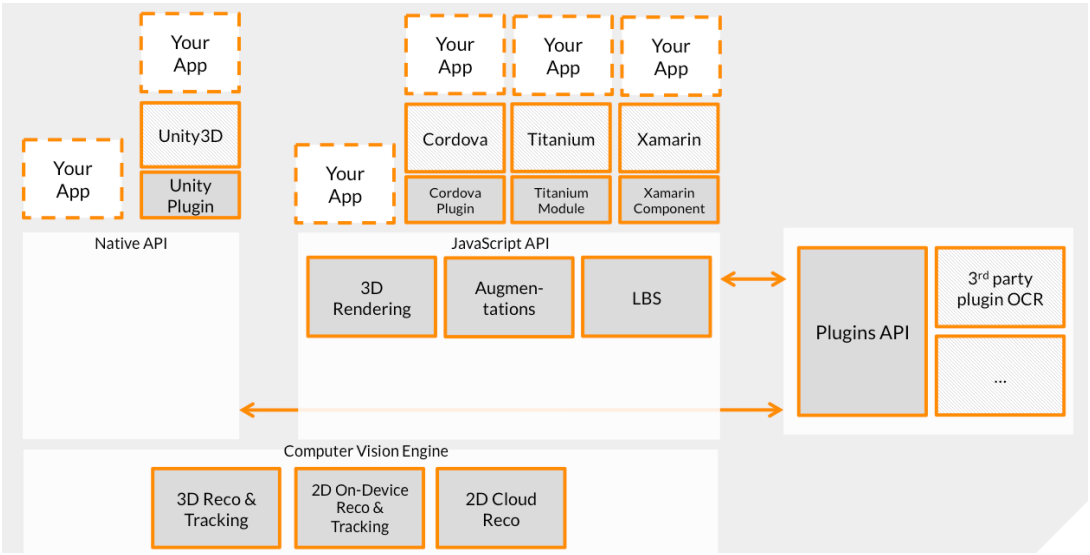


Figura 20: Arquitectura WIKITUDE (5.0)

2.9 Otras librerías

2.9.1 SharedPreferences

Sharepreferences- o preferencias compartidas, pertenece a la librería *Android.content* y nos permite almacenar las preferencias del usuario, en nuestro caso la localización, el nombre del usuario, el tipo de usuario, y otras características, en nuestra aplicación sin necesidad de utilizar otras herramientas que pueden llegar a ser más complicadas de utilizar [39].

Es una alternativa a utilizar SQLite para guardar este tipo de preferencias, aunque Android ha creado este método precisamente para administrar este tipo de datos. Cada preferencia se guarda en forma clave-valor, por lo que a cada una se le asignará una clave o un identificador, por la que será invocada para guardar, cambiar o eliminar el valor contenido en ella.

Cuando se crean las preferencias compartidas se crea un fichero XML con todas las claves-valores que vamos a introducir. Además, una aplicación puede crear varios

ficheros de preferencias compartidas y dependiendo del modo de acceso, estas pueden ser compartidas a otras aplicaciones. Existen 3 modos diferentes:

- MODE_PRIVATE. Sólo nuestra aplicación tiene acceso a estas preferencias.
- MODE_WORLD_READABLE. Todas las aplicaciones pueden leer estas preferencias, pero sólo la nuestra puede modificarlas.
- MODE_WORLD_WRITABLE. Todas las aplicaciones pueden leer y modificar estas preferencias.

2.9.2 Google maps API

Google proporciona una librería a los desarrolladores para poder implementar la vista de un mapa de tipo *Google maps* en los proyectos que trabajen con Android [40].

Para utilizar la librería debemos registrarnos como desarrolladores y obtener una clave para poder usarla. Una vez realizado estos pasos podremos importar la librería al proyecto.

Debemos tener un XML con el contenedor del mapa, y una clase que implemente las llamadas que puede realizar la librería. Por ejemplo la clase debe ser capaz de capturar la llamada *onMapReady* para la inicialización del mapa.

En este caso hemos tenido en cuenta la explicación dada por Navneet [41] del blog *androidtutorialpoint.com* para crear una vista calculando las distancias entre 2 puntos.

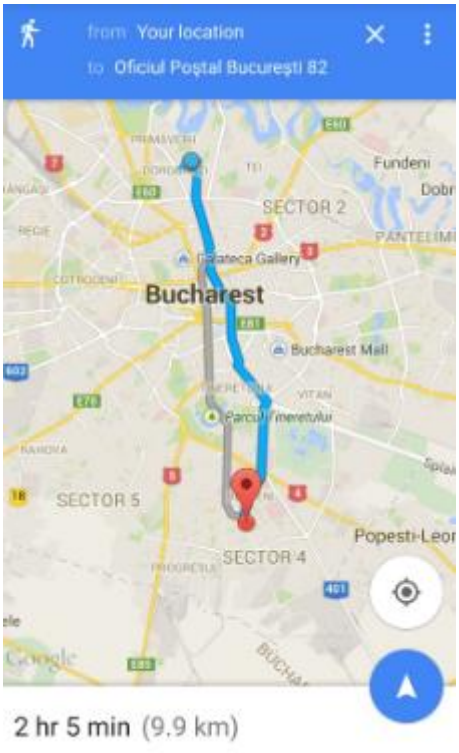


Figura 21: Captura de pantalla de la aplicación Google Maps

2.10 Elección de librerías a usar

En primer lugar antes de decidir que versión de Android vamos a utilizar, vamos a decidir qué tipo servidor escogeremos para guardar los datos de los usuarios, y las tablas que necesitamos.

Tenemos 3 opciones diferentes: la opción de crear un servicio web local, un servicio web externo, y un servicio Cloud. Hemos visto que las ventajas de un servicio Cloud son muchas, pero al ser una tecnología en la que es necesario pagar y que además podemos sustituir muchas de sus funciones con tecnologías locales, hace que descartemos este tipo de servicios. Los servicios Hosting tradicionales nos ofrecen muchas posibilidades, pero se quedan cortos en cuestión de espacio y rendimiento. La última opción, y la más conveniente para nuestros intereses, es un servicio web local, que nos dará un soporte mediante un API RESTFUL para obtener los datos que necesarios.

Para ello instalaremos XAMPP, de esta forma crearemos el servidor local e instalaremos MariaDB para la creación de la bases de datos. Crearemos entonces el servicio API con una serie de archivos (que se describirán en la siguiente sección) que nos permitirán conectarnos al base de datos desde fuera del servidor. Este servicio utilizará la tecnología PHP y podrá responder a solicitudes POST y GET, devolviendo una respuesta en formato JSON.

Por lo tanto JSON es la tecnología que utilizaremos para el intercambio de mensajes, ya que es una tecnología ligera y sencilla, características que necesitamos para los tipos de mensajes que vamos a enviar y devolver: mensajes de una sola temática (torneos, equipos o reglas de un usuario específico), por lo que complejidad de las operaciones no es alta.

Una vez hemos determinado que estructura tiene nuestro servicio web, pasaremos a seleccionar la versión de Android más adecuada para desarrollar nuestra aplicación.

Hemos visto que tanto la versiones *Jelly Bean* y *Lollipop* se reparten la mayor parte de los dispositivos donde Android es el sistema operativo instalado, pero podemos apreciar que *Lolilipop*, además de ser una versión más actual, nos permite una mejor integración y funcionalidad de la realidad aumentada, el reconocimiento de voz y la integración del paradigma de diseño Google Materials Design, por ello vamos a desarrollar la aplicación con una tecnología enfocada a dispositivos con una versión *Lollipop* o superior.

Vista la versión que utilizaremos para programar la aplicación vamos a pasar a elegir la base de datos y las estructuras que utilizaremos para guardar los datos del usuario dentro de la propia aplicación.

Tenemos dos opciones: guardar todos los datos en SQLlite o utilizar SQLite y SharePreference de forma conjunta. Por eficiencia y porque Android recomienda la utilización de las preferencias compartidas, vamos a utilizar las preferencias compartidas para guardar los datos básicos del usuario, y utilizaremos SQLite para guardar los datos de equipos, torneos y reglas que el usuario marque como favoritos.

Pasamos ahora a la elección del tipo de conexiones que vamos a realizar para intercambiar mensajes, esto es, las peticiones que nuestra aplicación hará a nuestro servidor externo.

La aplicación pasará la información del usuario así como los torneos, equipos y reglas favoritos, y el servicio web guardará esta información en la base de datos externa. Además tendrá que devolver, cuando la aplicación se lo pida, estos mismos datos. Estos datos pueden ser muy extensos, porque el servidor devuelve un resultado con una estructura variable, dependiendo del número de entradas devuelta por la base de datos. Por ello la tecnología *Volley*, aunque es la mejor opción en muchas ocasiones, en esta parte donde las respuestas pueden ser largas, no la vamos a utilizar. Por ello nos decantaremos por *Retrofit*, para el intercambio de mensajes.

Hemos descartado *HttpURLConnection* ya que es una opción desfasada y que no incorpora las facilidades que tanto Volley y Retrofit añaden.

Por otra parte debemos de escoger los tipos de conexiones que debemos hacer cuando nuestra aplicación busque los datos en la página web externa (*ultimatecentral.com*). En este caso Volley será necesario, ya que el contenido de la página suele ser estático con un máximo de resultados, por lo que la devolución no es muy grande y el formato es de texto. De esta forma tendremos los resultados en un mensaje de texto, con un formato HTML, los cuales debemos parsear para poder leerlos. Utilizaremos JSOUP para el parseo del texto HTML.

Ya hemos definido la base de la aplicación, es turno de escoger que tipo de tecnologías de reconocimiento y síntesis de voz vamos a utilizar.

Hemos descrito varias tecnologías pero por potencia, desarrollo y fácil implementación la tecnología que vamos a implementar para el reconocimiento de voz será *Android.speech*, ya que nos proporciona las suficientes funcionalidades para la tarea que vamos a realizar.

Por las mismas razones escogeremos *Android.speech.tts* para la síntesis de voz. Aunque en este punto es necesario recalcar la activación del propio sistema de accesibilidad de Android para aumentar las prestaciones de la aplicación. Se ha escogido la propia accesibilidad de Android, pero se puede escoger el asistente que mejor se adapte a las preferencias de los usuarios.

Otro punto que debemos escoger son las librerías de realidad aumentada que vamos a incorporar en la aplicación.

Como nuestro proyecto está orientado al uso de realidad aumentada a través de la geolocalización las librerías ARTOOLKIT y VUFORIA quedan descartadas.

Según las opciones que tenemos LAYAR es una buena opción para la geolocalización, ya que su código se integra fácilmente a la plataforma y existe mucha documentación acerca del uso y funcionamiento de la misma, sin embargo no es gratuita, lo que dificulta su puesta en marcha.

La decisión más acertada es WIKITUDE que aunque su versión gratuita es limitada y presenta una serie de inconvenientes, es mucho más que suficiente para las tareas que vamos a realizar. Además incluye una gran cantidad de documentación, ejemplos y uso de la librería.

El único inconveniente que presenta es la versión nativa. Esta versión está aún en fase de desarrollo, por lo que debemos utilizar la versión que integra JavaScipt, HTML, CSS y Jquery para la implantación de la RA en nuestra aplicación. La versión no nativa puede llamar a Actividades nativas, de esta forma se hace más fácil su integración.

Por último incorporaremos la librería *Google maps* para dar más atractivo y funcionalidades a nuestra aplicación.

2.11 Conclusiones

Podemos concluir que las alternativas para crear esta aplicación son muy variadas y que muchas veces pueden complementarse entre ellas para dar más robustez y potencia al resultado final.

También hemos descubierto que aún con el avance de la tecnología actual, existen muchas dificultades y limitaciones para desarrollar aplicaciones que se basen en la interacción del dispositivo ya sea con una persona física o con la realidad, sin depender de las interacciones tradicionales.

Capítulo 3

Análisis del sistema

En este capítulo se realiza un análisis de los requisitos de nuestra aplicación. Para ello vamos a analizar los requisitos funcionales y los requisitos no funcionales, además de presentar los distintos casos de uso que tiene la aplicación.

El análisis es una parte muy importante ya que nos permite distinguir las muchas funcionalidades del sistema, y nos permite poder distinguir que puede y que no puede realizar.

Para el análisis del sistema se tuvo en cuenta la encuesta realizada a jugadores de Ultimate Frisbee y que se encuentra en el anexo.

3.1 Requisitos

Vamos a pasar a describir los requisitos de nuestra aplicación.

3.1.1 Requisitos funcionales

- **Registro de usuario:** Permite la creación de un usuario que será guardado en la base de datos externa. Tendrá que comunicarse con el servidor externo.
- **Inicio de sesión:** Permite acceder a las distintas páginas de la aplicación una vez el usuario está registrado.
- **Buscar torneos:** Permite la búsqueda de torneos por nombre o por localización.
- **Buscar equipos:** Permite la búsqueda de equipos por nombre o por localización.
- **Visualización reglas:** Muestra las reglas del deporte.
- **Búsqueda reglas:** Permite la búsqueda por términos en las reglas.

- **Seleccionar equipos, torneos o reglas:** Permite acceder a la información referente a una regla, equipo o torneo seleccionado.

- **Visualización de favoritos:** Se debe poder ver un listado con los equipos, reglas y torneos favoritos.

- **Añadir favoritos:** Permite añadir torneos, equipos o reglas a favoritos.

- **Eliminar favoritos:** Permite eliminar torneos, equipos o reglas de favoritos.

- **Actualizar favoritos:** Permite actualizar los favoritos a la base de datos del servidor externo.

- **Escuchar información:** Permite escuchar la información relativa a una página en concreto.

- **Decir instrucciones:** Capta como entrada la voz con las instrucciones que la aplicación debe realizar.

- **Escuchar instrucciones:** Permite escuchar la información relativa a las instrucciones de una página en concreto.

- **Visualizar información del usuario:** Permite ver la información actual del usuario.

- **Cambiar ciudad:** Permite cambiar la información de la ciudad donde te encuentras.

- **Cambiar contraseña:** Permite cambiar la contraseña del usuario.

- **Seleccionar Realidad Aumentada (RA):** Permite seleccionar la opción de visualizar la realidad aumentada en equipos y torneos.

- **Seleccionar equipo o torneo en RA:** Selección de un torneo o equipo cuando se está en modo RA. El equipo o el torneo seleccionado debe estar muy diferenciado.

- **Detalles equipo o torneo RA:** Una vez seleccionado un elemento, permite ver los detalles concretos en lo referente a la localización geográfica.

- **Cerrar detalles equipo o torneo RA:** Permite el cierre de la ventana de detalles y la visualización del estado anterior a éste.

- **Visualización de ruta RA:** Una vez seleccionado permite ver la ruta desde el punto donde se encuentra el usuario al punto donde se encuentra el torneo o equipo seleccionado.

- **Indicaciones al punto RA seleccionado:** Permite abrir una aplicación externa que permite hacer de guía hasta el punto final seleccionado.

3.1.2 Requisitos no funcionales

La aplicación cumplirá con aquellos requisitos comunes para este tipo de aplicaciones:

Usabilidad: El tiempo de aprendizaje de la aplicación debe ser corto, además debe ser amigable y fácil de utilizar.

Optimización: Los resultados de las búsquedas y visualizaciones deben ser los más cortos posibles, es decir, el tiempo de respuesta debe ser corto.

Integración: La aplicación debe funcionar para todos los dispositivos Android que cumplen los requisitos de hardware y software.

Rendimiento: La aplicación debe ser fluida y proporcionar al usuario una experiencia correcta.

Disponibilidad: La aplicación debe ser fácil de descargar e instalar.

Fiabilidad: Se debe informar al usuario de los posibles fallos, y la aplicación debe hacer frente a los mismos.

Seguridad: No debe permitir la instalación de software malicioso en la aplicación.

Mantenibilidad: La aplicación debe proporcionar actualizaciones y debe ser mantenida.

3.2 Casos de uso

En los casos de uso podemos hacer una diferenciación de tres tipos:

- Caso de uso de la iteración normal: inicio sesión, registro, búsqueda e información (Figura 22).
- Caso de uso relativo al sistema de modulo del habla: escuchar u decir instrucciones, y escuchar la información contenida en una página (Figura 23).
- Caso de uso relativo a la realidad aumentada: visualizar elementos en AR y ver la ruta específica hacia un punto (Figura 24).

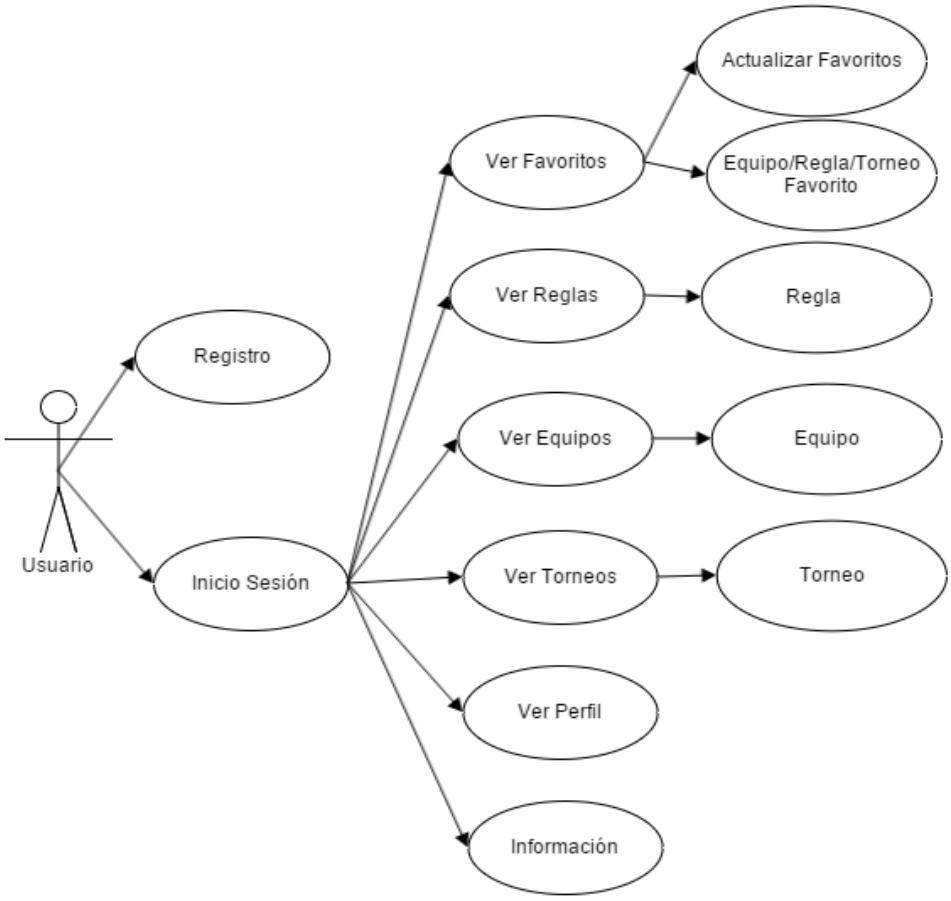


Figura 22: Caso de uso general

- Registro: El usuario puede registrarse en la aplicación.

- Inicio de sesión: El usuario puede entrar en la aplicación, da paso a las demás funcionalidades de la aplicación.

- Ver favoritos: El usuario puede visualizar las reglas, torneos y equipos favoritos. En esta funcionalidad podemos seleccionar un equipo, torneo o regla para visualizarlo, podemos actualizar los datos en el servidor externo, y podemos añadir o eliminar favoritos.

- Ver reglas: podemos visualizar las reglas del deporte, además de añadir o eliminar reglas a favoritos, ver una regla en concreto, y buscar términos.

- Ver torneos y equipos: podemos visualizar el resultado de la búsqueda por nombre o localización de equipos y torneos, además de añadir o eliminar equipos o torneos a favoritos, y ver un torneo o equipo en concreto.

- Ver perfil: podemos visualizar el perfil del usuario, además de poder cambiar la contraseña o ciudad del usuario.

- Información: información del autor de y propósito del sistema.

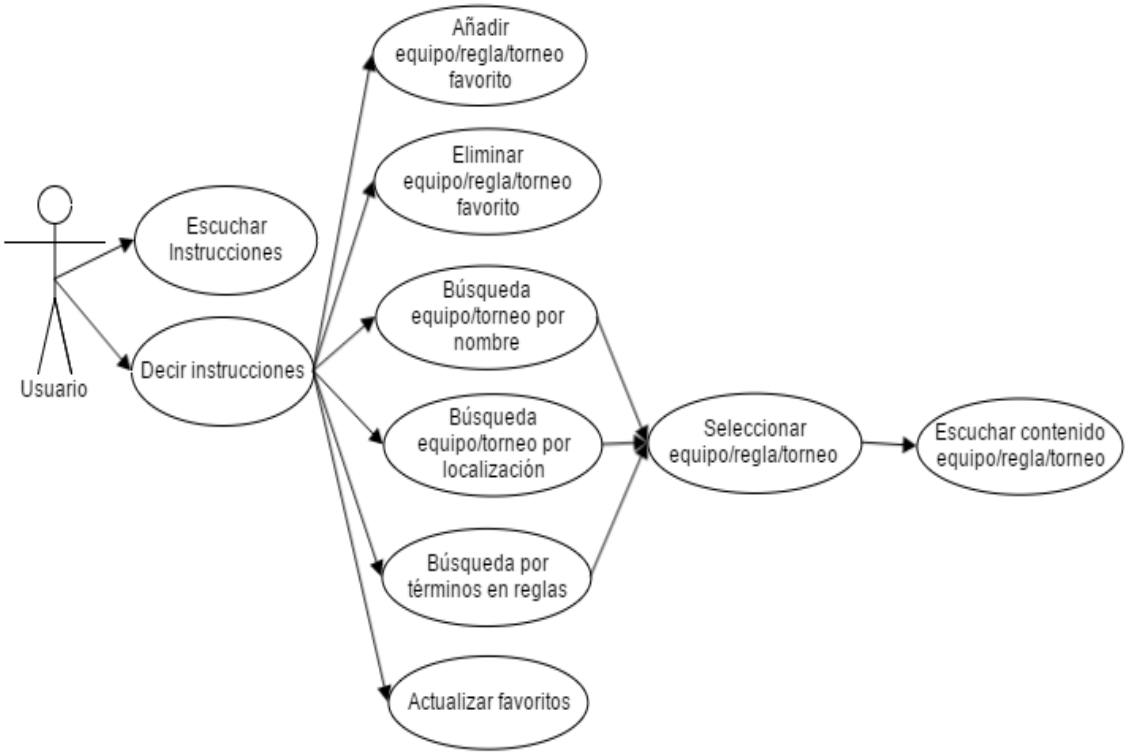


Figura 23: Caso de uso sistema de habla

- Escuchar instrucciones: El usuario puede escuchar las instrucciones de una página en concreto.

- Decir instrucciones: El usuario decir las instrucciones y el dispositivo captará y ejecutará dichas instrucciones.

- Añadir o eliminar equipos/reglas/torneos favoritos: El usuario puede añadir o eliminar favoritos siempre que se encuentre en las páginas de equipo, reglas, torneo o favoritos.

- Búsqueda de torneo/regla/equipo: El usuario puede buscar en equipos, torneos por localización, y por términos en reglas. Además la búsqueda muestra una lista donde se selecciona el elemento deseado y se visualiza el contenido.

- Escuchar contenido: Permite al usuario dar la instrucción de escuchar el contenido de la página en la que se encuentra.

- Actualizar favoritos: permite actualizar los favoritos de la base de datos externa.

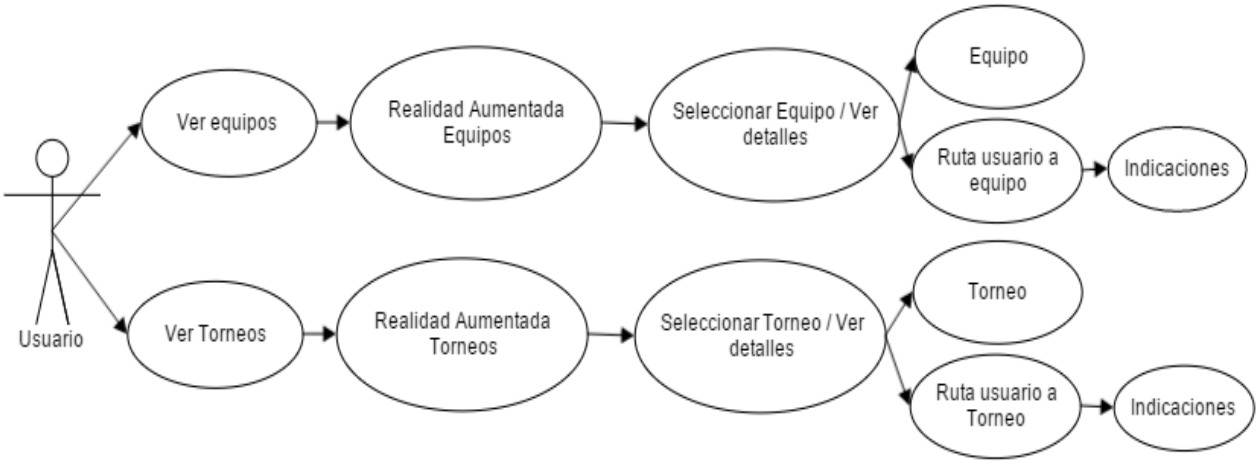


Figura 24: Caso de uso Realidad Aumentada

En este caso de uso solo se puede utilizar cuando estamos visualizando los equipos o los torneos.

- Ver torneos o equipos: Lista con los torneos o equipos buscados.
- Realidad aumentada: el usuario puede visualizar los puntos de interés con la ayuda de la cámara del dispositivo. Según de la dirección enfocada visualiza los torneos o equipos que se encuentran en el radio de distancia y en el sentido donde se enfoca.
- Seleccionar equipos o torneos, y ver detalles: El usuario puede seleccionar equipos y torneos cuando toca la pantalla donde se encuentra un punto de interés. Una vez seleccionado el usuario visualiza los detalles de dicho elemento.
- Equipo o torneo: muestra el contenido del equipo o torneo seleccionado.
- Ruta usuario a torneo o equipo: Muestra una página con la ruta desde el punto donde se encuentra el usuario hacia el punto de interés seleccionado (equipo o torneo).
- Indicaciones: Abre una aplicación externa (*Google maps*) donde el usuario puede visualizar la ruta con las indicaciones exactas para llegar hasta el punto indicado.

Capítulo 4

Diseño y descripción de los módulos del sistema

En este capítulo se mostrará el diseño detallado del sistema, presentando los diferentes módulos y las funcionalidades que realiza cada uno.

En primer lugar el nombre de la aplicación será *Upp*: juego de palabras entre *Up* y *App. Up* es la palabra que se grita cuando se lanza un disco en el deporte.

Ahora vamos a ver cómo está configurada nuestra aplicación en Android Studio, para poder ver como tenemos estructurado el proyecto:

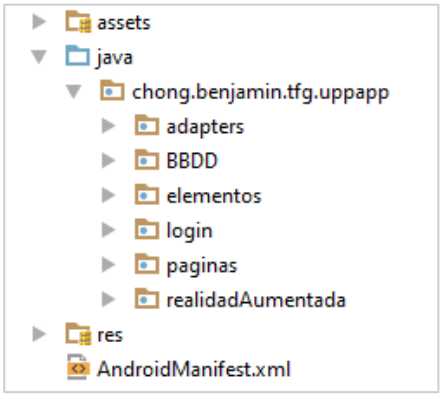


Figura 25: Estructura Principal Proyecto

Podemos observar que tenemos tres carpetas diferentes: *assets*, *java* y *res*.

La primera: *assets*, se guardan todos los elementos HTML, JavaScript y CSS que nos permitirán comunicar con la Realidad Aumentada.

La segunda: *java*, contiene todas las clases e interfaces Java que hemos creado para el proyecto.

La tercera: *res*, contiene los elementos XML necesarios para la visualización en el dispositivo Android.

En el directorio *java* se aprecia que existen seis carpetas diferentes. De esta forma podemos diferenciar más fácilmente que elementos se guardan en cada una y que parte representan:

- **Adapters:** En esta carpeta guardamos los adaptadores que se utilizarán para crear y guardar los elementos de las vistas que vamos a crear. Será utilizada por *Páginas* para creación de elementos.

- **BBDD:** Las clases y elementos de que vamos a utilizar para crear y gestionar la base de datos interna. La base de datos estará en permanente comunicación con diferentes partes del sistema.

- **Elementos:** Clases y elementos comunes que se utilizarán en las distintas partes del proyecto.

- **Login:** Clases para el inicio de sesión y el registros de usuarios.

- **Páginas:** Los diferentes *activities* de nuestro proyecto.

- **Realidad aumentada:** Diferentes clases que incluyen las clases para la utilización de realidad aumentada.

Por otra parte en el servidor externo tenemos los siguientes archivos y base de datos:

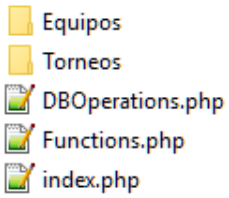


Figura 26: Carpetas alojadas en el servidor externo

- Index.php: Archivo que recibe las consultas GET o POST que llegan desde el dispositivo móvil. Se encarga de distinguir la tarea que debe realizar según la consulta recibida, y devolver la respuesta adecuada a esa consulta.

- Functions: Archivo que contiene distintas funciones invocadas por index.php que se utilizan para poder realizar las consultas y generar las respuestas dadas por la base de datos local. Es un archivo intermedio que comunica index.php con DBOperations, y viceversa.

- DBOperations: Archivo que contiene la información referente a la base de datos local: nombre de la base de datos, tablas, etc. Este archivo conecta con la base de datos local para consultar, extraer o insertar información en las tablas de la base de datos.

La carpeta Equipos y Torneos contienen un fichero *index* que permite devolver los torneos o equipos que se encuentran dentro del sistema según unas determinadas preferencias. Estas carpetas son utilizadas por el módulo Realidad Aumentada para obtener los torneos y equipos cercanos de la ciudad en la que nos encontremos.

Pasamos a ver las tablas que existen dentro de la base de datos:

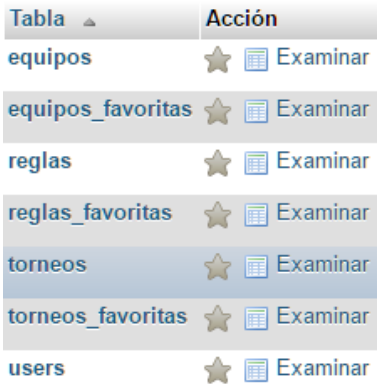


Figura 27: Tablas de la base de datos local

- Equipos: Información de los equipos.
- Equipos_favoritas: tabla que guarda los equipos favoritos de los usuarios. Hay una relación equipo/usuario.
- Reglas: Información de las reglas.
- Reglas_favoritas: tabla que guarda las reglas favoritas de los usuarios. Hay una relación regla/usuario.
- Torneos: Información de los torneos.
- Torneos_favoritas: tabla que guarda los torneos favoritos de los usuarios. Hay una relación torneo/usuario.
- Users: Tabla con la información relativa al usuario: nombre, ciudad, contraseña, etc.

Una vez vistos la estructura del sistema vamos a pasar a hablar de los módulos por lo que la hemos dividido.

Hemos visto que habíamos divido el proyecto en tres módulos diferentes: módulo base, módulo asistente sonoro y módulo realidad aumentada.

El módulo base correspondería a las carpetas. *Login, BBDD, elementos, adapter y páginas.*

El módulo asistente oral se encontraría en todas las carpetas de la aplicación, excepto *BBDD* y elementos, ya que este módulo debe actuar en cualquier momento, sin embargo, este módulo tiene funciones específicas que se repetirán a lo largo del sistema, i.e. funciones que utilizan una estructura común independientemente de las clases en la que se encuentre pero en las que cambia el contenido de la respuesta.

El módulo Realidad Aumentada le corresponde la carpeta realidad aumentada, además de otra carpeta que va asociada a ella: *assets*, que incluye los archivos HTML, JavaScript y CSS para la visualización de la realidad aumentada.

Consideraciones a tener en cuenta en el diseño:

- Para ingresar, registrarse, buscar torneos y equipos y torneos debemos estar conectados a internet
- Cuando se inicia sesión se destruye las tablas (si es que existen) de favoritos y se crean unas nuevas conforme a la información recibida por el servidor.
- Los favoritos se guardan en tablas SQLite y las preferencias en *sharedPreferences*.
- La codificación es en UTF-8.
- En la base de datos del dispositivo móvil, no se guardan los datos del usuario, ni en las tablas se crean una relación de tabla/usuario porque es la misma aplicación Android la encargada de diferenciar las tablas que pertenecen a un usuario u a otro.
- La aplicación se comunica con 2 tipos de enlaces diferentes: WEB SCRAPING y consultas a servidor externo. En el servidor se guardan los usuarios y sus preferencias, mientras que con WEB SCRAPING solo leemos los torneos y equipos que existen y los guardamos o actualizamos en el servidor cuando se guardan o se actualizan las preferencias del usuario.
- Las longitudes y latitudes son introducidas manualmente en la base de datos, ya que los datos que se recogen de la página web no contiene esta información, y la API de la web está aún muy poco avanzada.
- La sesión no se cierra hasta que se pulsa *Salir*.
- Los permisos que deben ser activados son: ubicación e internet.

4.1 Módulo base

El módulo base nos permite el registro, el inicio de sesión, la búsqueda de equipos y torneos, y la visualización de las reglas, el perfil del usuario y la información referente a la aplicación, además de la opción de salir de la misma.

En este apartado vamos a ver funcionalidad por funcionalidad que recursos y bibliotecas se han utilizado para poder interactuar con el módulo base.

En primer lugar debemos decir que la clase inicial de nuestra aplicación se llama *MainActivityUpp.java* y se encuentra dentro de la carpeta *login*. Esta actividad llama carga los *fragments* de inicio de sesión o de registro, según la acción que realicemos.

Una vez iniciada la sesión, la aplicación la mantiene todo el tiempo hasta que el usuario decide salir de la sesión. Distinguimos salir de la sesión actual, de salir de la aplicación ya que si se sale de la aplicación el sistema mantiene la sesión abierta.

4.1.1 Modulo Registro

Lleva a cabo el registro del usuario en la base de datos externa, que se encuentra en el servidor local. Las clases que hacen posibles el registro del usuario se encuentran en la carpeta *Login*, y se llama: *RegitroFragment.java*

En Registro el usuario debe de ingresar los campos: nombre, ciudad, email y contraseña, y luego iniciar la acción **registrar.**

El servidor utilizará el email para crear un identificador único que identifique a los usuarios. La contraseña será una contraseña alfanumérica que será codificada mediante un algoritmo *hash* y guardada en el servidor local.

El campo ciudad es una preferencia que permite cargar automáticamente equipos y torneos cuando se está en estas páginas.

Cuando todos los campos del registro están completos, el botón registro captura el evento e inicia la comunicación con el servidor para guardar los datos del nuevo usuario.

Existe además la posibilidad de ir a inicio de sesión al hacer un click en *Login Now*.

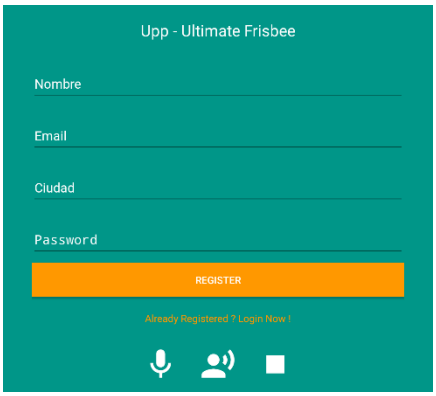


Figura 28: Captura de pantalla de la página de Registro

Una vez expuesto cómo se comporta esta página vamos a ver qué librerías utiliza:

En primer lugar, para la comunicación entre la aplicación y el servidor, *Registro* utiliza Retrofit para enviar los datos al servidor externo.

Retrofit crea un constructor para pasar la dirección URL donde se aloja las funciones que leerán las consulta de tipo Registro y que retornaran el mensaje de usuario ingresado o no ingresado. El siguiente parámetro es el tipo de conversor que vamos a utilizar, en este caso es un JSON, ya que la pregunta va codificada en este formato.

```
Retrofit retrofit = new Retrofit.Builder()
        .baseUrl(Constantes.BASE_URL)
        .addConverterFactory(GsonConverterFactory.create())
        .build();
```

Figura 29: Creación del objeto Retrofit con Retrofit builder

Una vez hemos puesto los parámetros necesarios vamos a crear la interfaz que utilizaremos para la comunicación entre servidor y aplicación. Esta interfaz es una interfaz común para todas las clases que quieren comunicarse con el servidor, cuyo nombre es: *RequestInterface.class*.

La clase contiene el tipo de consulta que se va a realizar, además de la ruta específica dentro del servidor, y por último declara y guarda los tipos de llamadas que se van a realizar en el servidor.

```
@POST("api.upp/")
Call<ServerResponse> operation(@Body ServerRequest request);
```

Figura 30: RequestInterface para las llamadas al servidor

Ya sabemos que contiene la interfaz, ahora vamos a instanciar nuestro objeto Retrofit a partir de esta clase:

```
RequestInterface requestInterface = retrofit.create(RequestInterface.class);
```

Figura 31: Creación de la interface con la función Retrofit Create

Ya tenemos nuestra clase Retrofit creada, nos falta las clases que serán leídas por el servidor y las clases que utilizaremos para guardar las respuestas recibidas del mismo. Para ello tenemos las clases: *user.java, ServerRequest.java y ServerResponse.java*.

Estas tres clases serán utilizadas para el intercambio de mensajes entre el servidor y la aplicación:

User.java: Clase común que se utilizará para guardar en un momento dado las propiedades y preferencias del usuario, tales como: nombre, email, ciudad, las reglas favoritas, los torneos favoritos y los equipos favoritos. Es una clase con campos fijos ya que el servidor los utiliza para la lectura de los datos del usuario, es decir, si queremos cambiar la estructura de esta clase debemos cambiar la lectura de la misma en el servidor externo.

ServerRequest.java: Esta clase es el mensaje que se envía desde la aplicación al servidor. Contiene el *user.java* con los datos de usuario y el tipo de operación que vamos a realizar: registro, leer torneos, leer equipo o leer reglas son algunas de estas operaciones.

ServerResponse.java: Esta clase se utiliza para la lectura de la respuesta dada por el servidor. Contiene un mensaje que es el resultado de la operación, una clase *user.java* para guardar los datos del usuario, además de los *arrays* de equipos, reglas y torneos.

Según la repuesta algunos campos estarán vacíos, por ejemplo, si vamos a registrar un nuevo usuario el mensaje de respuesta solo contendrá el resultado de la operación y los demás campos estarán vacíos.

Creamos entonces un objeto de tipo *user* para guardar temporalmente los datos del usuario, además creamos un objeto de tipo *serverRequest* para enviar la consulta y añadimos el tipo de consulta, en este caso *register*, y añadimos el usuario creado.

Una vez definidos los contenidos de los mensajes y los mensajes en sí vamos a pasar a realizar las llamadas al servidor.

```
Call<ServerResponse> response = requestInterface.operation(request);

response.enqueue(new Callback<ServerResponse>() {
    @Override
    public void onResponse(Call<ServerResponse> call, retrofit2.Response<ServerResponse> response) {

        ServerResponse resp = response.body();
        Snackbar.make(getView(), resp.getMessage(), Snackbar.LENGTH_LONG).show();
        progress.setVisibility(View.INVISIBLE);
        goToLogin();
    }

    @Override
    public void onFailure(Call<ServerResponse> call, Throwable t) {

        progress.setVisibility(View.INVISIBLE);
        Log.d(Constantes.TAG,"failed");
        Snackbar.make(getView(), t.getLocalizedMessage(), Snackbar.LENGTH_LONG).show();


    }
});
```

Figura 32: Código de la llamada al servidor local

Vemos que la respuesta que vamos a recibir es de tipo *serverResponse*, el cual captura mediante los métodos *OnResponse* y *OnFailure* las dos opciones posibles al ejecutar la llamada al servidor.

OnResponse captura el evento cuando la aplicación ha tenido éxito al comunicarse con el servidor. Si es así muestra el mensaje correspondiente a la llamada, es decir, si el usuario ha sido registrado o ya existe el usuario que hemos introducido. Después se dirige a la página de inicio de sesión.

OnFailure captuta evento cuando ha ocurrido un error en la comunicación entre la aplicación y el servidor. Devuelve el mensaje de error y muestra por pantalla el error ocurrido.

4.1.2 Módulo Inicio de Sesión

En inicio de sesión es la página por la que ingresaremos a las demás funcionalidades de la aplicación: Ver equipos, torneos, reglas, información y perfil. La clase que hace posible el inicio de sesión en la carpeta *Login*, y se llama *LoginFragment.java*

En primer lugar, para poder ingresar debemos estar registrados en el sistema (4.1.1), y debemos tener acceso a Internet, ya que de otra forma no podremos verificar la identidad del usuario.

Una vez cumplidos los requisitos, en esta página tenemos dos campos a rellenar: Email y contraseña.

Cuando introducimos el email y la contraseña tenemos la opción de entrar mediante la acción **Login**. Si es correcto guardará los datos del usuario en la aplicación e iniciara la carga de las tablas de reglas, equipos y torneos favoritos. Si no es correcto se mostrará

un mensaje de error. También tenemos la opción de ir a Registro desde inicio de sesión mediante la acción *Register Now!*

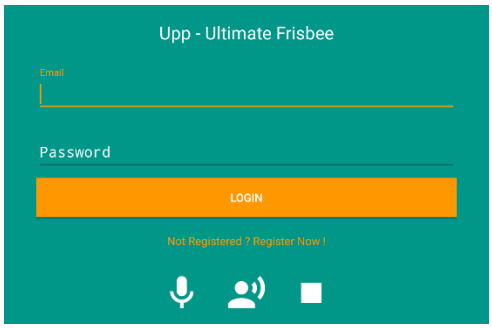


Figura 33: : Captura de pantalla de la página de Inicio de Sesión

Como ocurre en Registro, inicio de sesión utiliza Retrofit para poder comunicarse con el servidor y así verificar el usuario y la contraseña introducidos. Por ello, creamos un objeto Retrofit y lo inicializamos con el tipo de mensaje que se va enviar (mensaje de tipo *login)*, con los datos del usuario, y con las llamadas que vamos a realizar.

Cuando se captura el evento *onResponse* y el resultado es óptimo (cuando el servidor devuelve un mensaje de éxito) se van a guardan los datos del usuario con la ayuda de SharePreferences.

SharePreferences se utiliza para guardar datos de tipo clave-valor y poder acceder a ellos a lo largo de la vida la aplicación, por lo que resulta muy conveniente para acceder a las preferencias del usuario sin necesidad de utilizar SQLite.

En primer lugar debemos tener declarado en la clase un objeto de tipo SharePreference e inicializarlo:

```
private SharedPreferences pref;
pref = getActivity().getSharedPreferences("MisPreferencias", Context.MODE_PRIVATE);
```

Figura 34: Inicialización SharedPreferences

Cuando hemos inicializado estas preferencias podemos editar y guardar los datos que queremos:

```
SharedPreferences.Editor editor = pref.edit();
editor.putBoolean(Constantes.IS_LOGGED_IN,true);
editor.putString(Constantes.EMAIL,resp.getUser().getEmail());
editor.putString(Constantes.NAME,resp.getUser().getName());
editor.putString(Constantes.UNIQUE_ID,resp.getUser().getUnique_id());
editor.putString(Constantes.TIPO_USUARIO,resp.getUser().getTipoUsuario());
editor.putString(Constantes.CITY,resp.getUser().getCity());
editor.apply();
```

Figura 35: Edición del objeto SharePreferences en la página Login

Podemos ver que guardamos todo tipo de datos: un booleano para saber si el usuario ha iniciado la sesión, un string por cada campo que tenemos: ciudad, nombre, email, unique_id (id que se crea en la base de datos).

Ahora que tenemos todos los datos disponibles vamos a pasar a cargar las tablas de favoritos del usuario. Para ello hemos creado una página intermedia que se encargará de

comunicarse con el servidor y extraer la información referente a los torneos, equipos y reglas favoritas; para luego guárdalo en la base de datos de la aplicación.

Esta clase se llama *transition_login.class* y al igual que las anteriores utilizará Retrofit para enviar los mensajes de consulta y recibir los mensajes de respuesta. El mensaje de consulta en este caso será obtener los equipos, reglas y torneos favoritos, y el mensaje de respuesta, si la consulta ha sido satisfactoria, debe contener las reglas, equipos y torneos favoritos.

En esta clase cuando se captura el evento *onResponse* se debe verificar si el mensaje de respuesta es exitoso. Si es así, se debe guardar los datos en la base de datos de la aplicación con la ayuda de SQLite.

La base de datos que se ha creado dentro de la aplicación guarda los equipos, reglas y torneos favoritos, por lo que las tablas que existen dentro de ellas se llaman: equipos, reglas y torneos.

Las clases que se utilizan para crear y manipular la base de datos son: *UppBBDD.java* y *UppDBHelper.java*. La primera contiene los nombres y atributos de las tablas y la segunda contiene el nombre de la base de datos, funciones de creación de las tablas, y las funciones de inserción de elementos dentro de la misma.

```
sqLiteDatabase.execSQL("CREATE TABLE " + UppBBDD.EquiposBBDD.TABLE_NAME + " ("
        + UppBBDD.EquiposBBDD._ID + " INTEGER PRIMARY KEY AUTOINCREMENT,"
        + UppBBDD.EquiposBBDD.ID + " VARCHAR(100) NOT NULL,"
        + UppBBDD.EquiposBBDD.DESCRIPCION + " VARCHAR(256) DEFAULT '',"
        + UppBBDD.EquiposBBDD.TORNEOS + " TEXT DEFAULT '',"
        + UppBBDD.EquiposBBDD.JUGADORES + " TEXT DEFAULT '',"
        + UppBBDD.EquiposBBDD.URL_IMG + " VARCHAR(256) DEFAULT '',"
        + UppBBDD.EquiposBBDD.POS_IMG + " VARCHAR(256)  DEFAULT '',"
        + "UNIQUE (" + UppBBDD.EquiposBBDD.ID + "))");
```

Figura 36: Ejemplo tabla SQLite en la aplicación

Para introducir los datos recibidos dentro de las tablas primero se debe crear la base de datos (si la base de datos no existe), y eliminar todos los datos existentes en ella:

```
///vamos a crear la base de datos si esta no existe
UppDBHelper usdbh = new UppDBHelper(transition_login.this);

SQLiteDatabase db = usdbh.getWritableDatabase();

//borramos los registros de la tabla reglas para luego actualizarla
db.delete( UppBBDD.ReglasBBDD.TABLE_NAME,null,null);
db.delete( UppBBDD.EquiposBBDD.TABLE_NAME,null,null);
db.delete( UppBBDD.TorneosBBDD.TABLE_NAME,null,null);
```

Figura 37: SQLite inicialización de las tablas de la base de datos interna de la aplicación

Si la base de datos se crea y además los torneos, equipos y reglas de la consulta no están vacíos, entonces se introduce dentro de la base de datos.

Para ello se recorre cada array de torneo, equipo o regla recibido, se coge su identificador y se convierte a UTF-8 para evitar fallos de codificación. Luego se debe añadir cada elemento transformado a la tabla correspondiente (Figura 38).

```
for (int i_2 = 0; i_2 < tanano; i_2++) {
    try{
        String b = resp.getEquipos()[i_2].getEquipo_id();
        String a = URLDecoder.decode(b,"utf-8");
        resp.getEquipos()[i_2].setEquipo_id(a); ;
    }
    catch (UnsupportedEncodingException error){

    }
    db.insert(
            UppBBDD.EquiposBBDD.TABLE_NAME,
            null,
            resp.getEquipos()[i_2].toContentValuesEquipo()
    );
}
```

Figura 38: Ejemplo para añadir equipos en la tablas de la base de datos de SQLite

Cuando se ha introducido todos los datos recibidos y además no ha existido ningún tipo de fallo, entonces se puede pasar a la siguiente actividad: *favoritos.java*

Si durante la ejecución de ha existo fallos, se indica al usuario el tipo de fallo que ha ocurrido y se pasa la siguiente actividad.

Por último se puede observar que las tablas solo llevan la información relativa a los elementos creados, no llevan información alguna de los usuarios, esto se debe a que es el mismo sistema el que se encarga de distinguir los usuarios que existen en la aplicación y cargar los elementos según este criterio.

Un usuario no podrá ver los datos de otro usuario si inicia sesión desde el mismo dispositivo, ya que la naturaleza misma de la aplicación impide la carga de otros datos que no sean los tuyos, además de borrar los datos anteriores que hayan sido almacenados.

4.1.3 Elementos comunes: Listas, favoritos y contenido

En este apartado veremos cómo se crean las listas que podemos ver en las páginas de reglas, torneos y equipos. Para ello vamos a presentar la creación básica de este tipo de listas. Las clases que se utilizan para mostrar el contenido se encuentra en *páginas* y se llaman *ReglaContenidoActivity.java*, *EquipoContenidoActivity.java* y *TorneoContenidoActivity.java*.

Creación del contenido

Para crear las listas se ha utilizado el componente de Android RecyclerView. Este componente es un contendor de elementos en forma de lista (**¡Error! No se encuentra el origen de la referencia.**), que nos permite manejar grandes cantidades de objetos que se actualizan constantemente, limitando la visibilidad de los mismos.

Este componte además nos permite configurar una serie de animaciones para la eliminación, desplazamiento y creación de elementos de la lista en tiempo real.

Para la creación y el mantenimiento del RecyclerView se necesita una fuente de datos que provea la información lógica de cada elemento, un adaptador (*adapter*) que los lea,

interprete e infle (los visualice en la pantalla), y un componente llamado LayoutManager que permite para añadir y reutilizar los *views* dentro del mismo Recycler.

LayoutManager se encarga de calcular las posiciones fuera del rango del usuario, para así reemplazar el contenido de un ítem fuera del volumen visual por el contenido de otro, es decir, se encarga de remplazar un ítem por otro cuando este no está visible. Además reduce los tiempos de ejecución con respecto a otros métodos más tradicionales

Para implementar este elemento se debe ubicar el siguiente código en la actividad donde queremos visualizar la lista, en este caso: *regla_content_reglas.xml, torneo_content.xml y equipo_content.xml.*

```
<android.support.v7.widget.RecyclerView
    android:id="@+id/recycler_card_view"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:scrollbars="vertical">

</android.support.v7.widget.RecyclerView>
```

Figura 39: Ejemplo de definición de un Recyclerview para las reglas

Una vez introducido el código la actividad (*ReglasActivity.java, EquipoActivity.java o TorneoActivity.java*) debe capturar el elemento y asignarle el *adapter* al LayaoutManager.

Antes de asignar el adaptador lo primero es crearlo, y para conseguir este objetivo se debe declarar la fuente de datos que se utilizará.

Para implementar la fuente de datos creamos las clases *Item_cards_regla.java, Item_cards_equipos.java y Item_cards_torneos.java*, los cuales contendrán los atributos que guardarán la información de cada elemento de la lista: nombre del elemento, descripción breve y la imagen que se mostrará en la lista (las imágenes se encuentran en la carpeta *drawable* del proyecto para las reglas, y las imágenes de los equipos y torneos se cargaran leyéndolas de la página de consulta).

Esta clase servirá para que el adaptador pueda crear los elementos dentro del RecyclerView.

El adaptador es pues el elemento principal que creará individualmente cada elemento que contendrá la lista. Para ello creamos en la carpeta *adapter* las clases *CardsAdpater.java, CardsAdpaterEquipos.java* y *CardsAdpater Torneos.java* que heredarán de la clase padre *RecyclerView.Adapter*.

Como se ha expuesto anteriormente el adaptador debe crear individualmente los elementos de la lista, para ello se crea una clase interna que extenderá de la clase *RecyclerView.ViewHolder*, y que representa un elemento de la lista, lo que permite inicializar los elementos y las acciones que van a realizar. La clase interna se llamará *MyViewHolder.*

Por otra parte, cuando se crea el adaptador se hace referencia que los elementos o ítems dentro del mismo son del tipo *MyViewHolder*. De esta forma *LayoutManager* sabe dónde dirigirse.

Vamos a pasar a elegir el contenedor que mostrará cada elemento dentro de la lista. El contenedor elegido es del tipo *cardView* ya que nos permite la utilización de elementos de diseño ideales para nuestro paradigma.

El *cardView* se debe declarar dentro del adaptador, en la clase interna, para de esta forma inicializar el objeto y mostrar el contenido.

Otro aspecto importante de los adaptadores son las funciones *onCreateMyViewHolder*() y *onBindViewHolder*().

La primera infla el contenido de un nuevo elemento o ítem para la lista. La segunda realiza las modificaciones del contenido de los ítems de la lista, es decir, el contenido de las *cardViews*.

```
@Override
public MyViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
    View itemView = LayoutInflater.from(parent.getContext())
            .inflate(R.layout.regla_card_list_row, parent, false);
    return new MyViewHolder(itemView);
}

@Override
public void onBindViewHolder(MyViewHolder holder, int position) {
    Item_cards_regla itemcardsregla = cardList.get(position);
    holder.imagen.setImageResource(itemcardsregla.getImagen());
    holder.nombre.setText("Regla " + (position+1) +": " + itemcardsregla.getNombre());
    holder.descr_breve.setText(itemcardsregla.get_descr_breve());
    holder.currentItem = cardList.get(position);
```

Figura 40: Ejemplo de inicialización de un Adaptador para la página de Reglas

Cuando se sabe que es lo que va a realizar cada elemento de la lista, como se va a distribuir dentro la misma y cual es contenido que se mostrará, se puede pasar a relacionar el adaptador creado con el RecyclerView de la página principal.

Para ello primero capturamos el RecyclerView declarado en el XML, luego creamos el adaptador con los elementos donde se encuentran las informaciones de los elementos de la lista (*ítems*: array de tipo *Item_cards_regla.java*), inicializamos el LayoutManager y asignamos el adaptador al RecyclerView.

En este momento la lista está vacía por lo que no se visualizará nada, por ello debemos añadir elementos *ítems* y actualizar el adaptador para mostrar los cambios.

Todo los contenidos de los elementos en reglas se encuentran dentro de la aplicación por lo que no hace falta acceder a una base de datos para crear los elementos, simplemente hace falta pasar la referencia cuando se crea un nuevo ítem.

En equipos y torneos debemos realizar las consultas para poder obtener los nombres, las descripciones y las imágenes que necesitamos.

```
recyclerView = (RecyclerView) findViewById(R.id.recycler_card_view);
mAdapter = new CardsAdpater(items,this);
RecyclerView.LayoutManager mLayoutManager = new LinearLayoutManager(getApplicationContext());
recyclerView.setLayoutManager(mLayoutManager);
recyclerView.setItemAnimator(new DefaultItemAnimator());
recyclerView.setHasFixedSize(true);
recyclerView.setAdapter(mAdapter);
items.add(new Item_cards_regla(R.drawable.timeout, "Tiempos Fuera", R.string.regla_def_21, R.string.reglas_timeout_b_d));
mAdapter.notifyDataSetChanged();
```

Figura 41: Inicialización RecyclerView en la página de reglas

Creación Botón Favorito

Un aspecto importante de la aplicación es la opción de poder añadir e eliminar favoritos. Podemos añadir o eliminar favoritos gracias a un botón en forma de corazón situado en la parte superior derecha de la *cardView*.

Cuando se añade un favorito el botón pasa a tener fondo naranja, mientras que si la regla, equipo o torneo no es favorito entonces el botón tiene solamente el exterior pintado de verde. De esta forma nos aseguramos mostrar cuando una regla, equipo o torneo esta seleccionada como favorito y cuando no.

Figura 42: Icono de Favoritos

Para poder crear el icono primero hemos guardado ambas imágenes en el *drawable* de la aplicación, para luego crear el botón dentro del *cardview* y asignar como imagen predeterminada la imagen del corazón no seleccionado.

Una vez está el diseño implementado, tenemos que ver cómo se comporta la aplicación internamente cuando se pulsa el botón, o cuando ya está el elemento seleccionado como favorito.

Primero debemos saber que las reglas, equipos o torneos favoritos están dentro de la base de datos, por lo que cuando creamos cada elemento de la lista debemos comparar los elementos que vamos creando con los de la base de datos. Si coinciden debemos cambiar el icono del elemento creado de no favorito (predeterminado) al icono de favorito.

Para poder verificar los favoritos al inicializar las listas, debemos capturar en el *onBindViewHolder* del adaptador las acciones y las verificaciones necesarias, de esta forma cambiarán individualmente los botones favoritos de cada elemento de la lista.

```
UppDBHelper usdbh = new UppDBHelper(mContext);
SQLiteDatabase db = usdbh.getWritableDatabase();

String[] campos = new String[] {UppBBDD.ReglasBBDD.ID,UppBBDD.ReglasBBDD.DESCRIPCION,
                                UppBBDD.ReglasBBDD.POS_IMG,UppBBDD.ReglasBBDD.URL_IMG};
String[] args = new String[] {holder.currentItem.getNombre()};

Cursor a = db.query(UppBBDD.ReglasBBDD.TABLE_NAME, campos, UppBBDD.ReglasBBDD.ID+"=?",
                    args , null, null, null);

if(db != null && a!=null && a.getCount()>0 ){
    holder.reglaNoFavorito.setActivated(true);
    db.close();
}
else{
    holder.reglaNoFavorito.setActivated(false);
    db.close();
}
```

Figura 43: Código para verificar que un elemento pertenece a los Favoritos

Cuando hemos inicializado los elementos y hemos comprobado los favoritos, entonces debemos capturar el paso de favorito a no favorito y viceversa.

Para ello, cuando creamos el *viewholder* en el adaptador capturamos el *listener* del botón favorito cuando se hace *click*, es decir, capturamos la acción cuando se pulsa el botón favorito. Si se pulsa y el botón está desactivado (no es favorito) entonces el botón se activa, cambia al icono favorito, y añade el elemento a la base de datos que le corresponde. Si se pulsa y el botón está activado (es favorito) entonces el botón se desactiva, cambia al icono no favorito, y elimina el elemento de la base de datos.

```
if(reglaNoFavorito.isActivated()){
    UppDBHelper usdbh = new UppDBHelper(mContext);
    SQLiteDatabase db = usdbh.getWritableDatabase();
    if(db != null){
        db.delete(UppBBDD.ReglasBBDD.TABLE_NAME, UppBBDD.ReglasBBDD.ID
        + "='" + currentItem.getNombre()+"'", null);
        db.close();
    }
    reglaNoFavorito.setActivated(false);
}
else{
    UppDBHelper usdbh = new UppDBHelper(mContext);
    SQLiteDatabase db = usdbh.getWritableDatabase();
    Regla regla = new Regla(currentItem.getNombre(),currentItem.get_text_context(),
                            currentItem.getImagen(),"");
    if(db != null){
        long inserecion = db.insert(
                UppBBDD.ReglasBBDD.TABLE_NAME,
                null,
                regla.toContentValuesRegla());
        db.close();
    }
    reglaNoFavorito.setActivated(true);
```

Figura 44: Código para activar y desactivar Favoritos en reglas

Mostrar reglas, equipos o torneos individualmente

En este apartado veremos que sucede cuando se pulsa en un ítem individual de la lista de reglas, equipos o torneos.

Cuando se pulsa en un ítem, es decir, en alguna *cardView* del *recyclerView* pasamos a una actividad que mostrará toda la información relacionada a ese ítem.

En esta actividad se mostrará la imagen asociada al ítem, su título y el todo el contenido de la regla, equipo o torneo.

Figura 45: Ejemplo del contenido de un Ítem de tipo Regla

4.1.4 Módulo Reglas

Es la página donde se visualiza en forma de lista ordenada las reglas que contiene el deporte. La clase se encuentra en *páginas* y se llama *ReglasActivity.java*.

En esta página se muestran las distintas reglas que existen, presentado cada una de ellas de forma individual para la interacción y visualización de las mismas.

Las reglas se muestran con un título principal acompañado de un pequeño párrafo debajo de ellas, además de un marcador para poder seleccionar o deseleccionar las reglas favoritas del usuario.

Una vez seleccionado la regla deseada se visualiza la información de la misma en una actividad paralela a la principal.

Por otra parte esta página da la opción de buscar términos, palabras o frases dentro de las reglas. Cuando se busca la página muestra las reglas que contienen coincidencias con la búsqueda. Para volver a cargar todas las reglas solo hace falta pulsar el botón de reglas.

A diferencia de los torneos y equipos, la información de las reglas se encuentra dentro de la misma aplicación, por lo que no hace falta conexión a Internet para interactuar con este apartado.

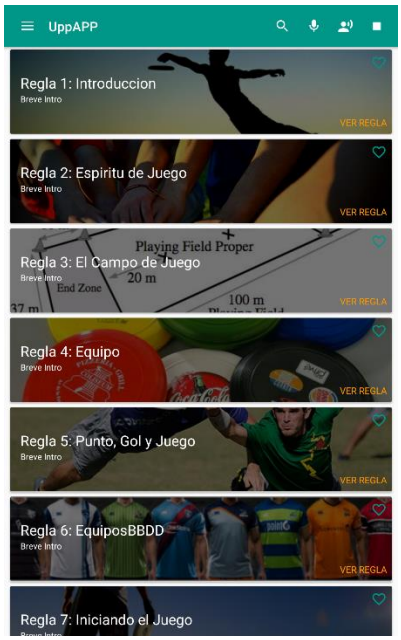


Figura 46: Captura de pantalla de la página Reglas

Hemos visto en el apartado anterior (4.1.3) como se crean los ítems de la lista, como se añaden o eliminan favoritos, y como se muestran los contenidos individuales de cada elemento.

Toda la información con respecto a las reglas, como los títulos, descripciones, contenido e imágenes se encuentran dentro de la aplicación, en la carpeta *res*/*drawables* y *res/values*, por lo que para crear los contenidos solo hace falta pasar la referencia al método o clases que se quiere mostrar.

Búsqueda de términos, palabras o frases: funciones.

En primer lugar, para poder acceder a la búsqueda hemos creado un botón que se encuentra en la barra superior de la aplicación con el símbolo de la lupa .

Una vez capturado el *listener* del botón podemos comprobar si el las palabras o frases que se han introducido se encuentran dentro de alguna regla, pero primero debemos borrar todas las reglas del adaptador que se encuentran visibles, y transformar a minúsculas la frase o palabra introducida.

Cuando borramos todas las reglas pasamos a comprobar la frase, recorriendo todas las reglas y verificando si está contenida con la función ***indexOf***. Si la búsqueda es exitosa entonces añadimos la regla al adaptador para luego mostrarlo. De esta forma tenemos una lista con las reglas que contienen las consultas.

4.1.5 Módulo Equipos

Vamos a ver ahora la página de equipos. Esta página muestra inicialmente los equipos que se encuentran en la ubicación que hemos introducido al registrarnos, además

nos permite acceder al módulo realidad aumentada que veremos más adelante. La clase se encuentra en *páginas* y se llama *EquipoActivity.java*.

Como hemos visto en Reglas, Equipos muestra una lista con los nombres de los equipos, un breve subtítulo y la opción añadir o eliminar favoritos.

El contenido de un ítem individual también es similar al de las reglas, ya que contiene la imagen del equipo, el título y la información relativa al equipo.

La principal diferencia con respecto al anterior apartado es que la información de los equipos no se encuentra dentro de la aplicación, si no se encuentra en una página externa llamada *ultimatecentral.com*

Otra diferencia importante es que la búsqueda se puede hacer por el nombre del equipo o por ubicación.

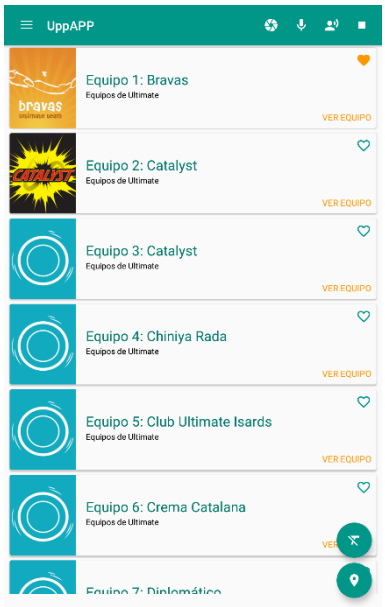


Figura 47: Captura de pantalla de la página Equipos

Búsqueda por nombre o ubicación

Para realizar este tipo de búsquedas hemos añadido dos botones diferentes:

Para búsquedas por nombres de equipo
Para búsquedas por ubicación

Al igual que con las reglas capturamos el evento del *listener* y transformamos la consulta que vamos a realizar a minúsculas y al formato que entiende las url de las páginas web. Por ejemplo: cambiar el espacio en blanco por el símbolo “+”. Una vez tenemos la consulta transformada vamos a comunicarnos con la página *ultimatecentral*.

Cuando realizamos la consulta la guardamos en una variable para poderla utilizar más veces en el futuro.

La URL que vamos a consultar varía según el tipo de búsqueda que realicemos:

- http://ultimatecentral.com/en_us/t/location para la búsqueda de por localización;
- http://ultimatecentral.com/en_us/t/search para la búsqueda por nombre.

Comunicación entre ultimateCentral.com y la aplicación

La aplicación debe comunicarse con la página, además de guardar y crear los ítems si la consulta es exitosa. Esta acción la realizamos en el adaptador ya que es aquí donde podemos crear y manipular los ítems de las listas.

En el adaptador de los equipos *CardsAdapterEquipos.java* creamos una función llamada ***leer_datos*** que recibirá por parámetros la consulta a realizar y el tipo de consulta (si es por nombre o por ubicación).

Para estar más seguros de la codificación codificamos la consulta con *URLEncoder* para luego verificar si se está realizando una consulta. Si no se está realizando alguna consulta vamos a generar la nuestra.

Para crear la consulta hemos utilizado la librería Volley, la cual crea una petición de tipo string ya que lo que devuelve la petición es un texto en formato HTML.

Cuando Volley crea la petición y la gestiona mediante una cola de peticiones. Si la respuesta de la petición se encuentra en caché entonces se parsea la respuesta y se muestra en el hilo principal, si no es así se envía a la cola de peticiones pendientes.

Las peticiones pendientes son ejecutadas en segundo plano por el componente llamado Network Dispatcher, el cual selecciona las peticiones y realiza las transacciones HTTP a la página externa.

En nuestro caso las repuesta que nos entrega Volley va ser en forma de string, por ello declaramos que la repuesta debe llevar este tipo de dato.

```
StringRequest request = new StringRequest(Request.Method.GET, consulta_url,
        new Response.Listener<String>() {
```

Figura 48: Ejemplo de petición Volley

Para poder capturar las respuestas tenemos la función **onResponse** y **onErrorResponse**. La primera nos dice si el servidor nos ha devuelto una repuesta, y la segunda si ha existido un error durante la comunicación.

Si la respuesta es correcta entonces pasaremos a parsear el String que hemos obtenido. Si no lo es mostrará un mensaje de error al usuario.

El parseo del texto lo hacemos con la librería JSOUP, la cual nos permite crear un documento HTML a partir de un texto plano, si este último tiene una estructura HTML.

Cuando tenemos el documento, lo podemos dividir en secciones según nos interesa. Como lo que nos interesa es encontrar los equipos encerrados, buscamos la palabra clave

de los contenedores de los equipos y extraemos su título, la URL de la página del equipo y la URL de la imagen del equipo, de esta forma tenemos la información necesaria para mostrar, por una parte, los elementos en las lista, y por otra, poder cargar el contenido de un ítem particular cuando se seleccione.

Por último debemos crear los ítems en el adaptador del equipo, para ello contamos con la clase *Item_cards_equipos.java*. Por cada equipo leído crearemos una clase de este tipo y la añadiremos al array de objetos de esta clase que está contenida en el adaptador.

```
public void onResponse(String response) {
    pos_query++;
    respuestaString= response;
    org.jsoup.nodes.Document doc = Jsoup.parse(response);
    String title = doc.title();

    Elements anchors = doc.select("div.span4.media-item-wrapper.spacer1");
    for (Element anchor : anchors) {
        Item_cards_equipos equipo ;
        //vamos leyendp cada post de la busqueda y guardamaos gracias a las query lo que queremo
        Element a = anchor.select("a").first();
        String url_a = a.attr("Style");
        //cogemos la url de la imagen
        url_a = url_a.replace("background-image: url('","");
        url_a = url_a.replace("')","");
        String url_equipo = a.attr("href");
        String url_generica = "http://ultimatecentral.com/en_us/t/";
        String[] parts = url_equipo.split("/");
        url_generica += parts[parts.length-1] + "?sso=1";
        String nombre = anchor.select("div h3").text();

        equipo = new Item_cards_equipos(nombre, "NO","Equipos de Ultimate",url_a,url_generica);
        cardList.add(equipo);
```

Figura 49: Código para la captura del evento OnResponse de Volley en Equipos

Una vez hemos leído todos los equipos de la página notificaremos al adaptador que existe nuevos equipos y que deben ser mostrados, de esta forma se muestran en el hilo principal los equipos recién leídos.

```
numero_items_anteriores = cardList.size();
notifyDataSetChanged();
leyendoDatos = false;
```

Figura 50: Ejemplo de notificación al adaptador cuando la lista cambia de tamaño

Hemos visto como crear la petición que utilizará volley, sin embargo no hemos creado la clase que encapsula las funciones necesarias para que funcione Volley. Por ello hemos creado un patrón Singleton llamado *SingletonEquipos.java.*

Esta última clase será utilizada por la actividad equipos para crear y manipular las consultas realizadas a la página externa. Además esta clase contiene la cola de peticiones y el contexto donde se ejecuta las peticiones, de esta forma sabe dónde debe presentar los datos obtenidos.

También contiene la propiedad *synchronized* para evitar bloqueos de acceso de los distintos hilos que se ejecutan.

Por último la clase contiene una instancia de un ImageLoader. Esta instancia es importante ya que sirve para cargar las imágenes que provienen de una URL. Por ello a esta instancia se debe asociar una cola de peticiones (para poder añadir la petición) y un objeto de tipo ImageCache para guardar la imagen cargada en la cache.

Para añadir la imagen al cardView, en la función **onBindViewHolder** se debe añadir las siguientes líneas:

```
NetworkImageView imagenPost = (NetworkImageView) holder.imagen;
ImageLoader imageLoader= SingletonEquipos.getInstance(mContext).getImageLoader();
imagenPost.setImageUrl(cardList.get(position).getUrl_imagen(), imageLoader);
```

Figura 51: Ejemplo de petición Volley para la obtención de una Imágen

Para añadir la petición agregamos la petición al Singleton. De esta forma se cargaran los ítems, para luego crear los cardViews en la función **onBindViewHolder**.

```
SingletonEquipos.getInstance(this.mContext).addToRequestQueue(request);
```

Figura 52: Ejemplo de petición Singleton para Volley

Por otra parte cuando realizamos las consultas solo podemos leer 14 equipos, por lo que si queremos mostrar más equipos debemos hacer un scroll hacia abajo.

Si hacemos un scroll hacia abajo y el número de equipos mostrados es 14 o más la aplicación hará otra consulta a la página para extraer los siguientes 14 equipos. Si no existen más equipos entonces la aplicación informa al usuario que no existen más.

Cuando realizamos una nueva consulta se borran del hilo principal todos los equipos mostrados y se empieza de nuevo con una nueva consulta.

Mostrar equipos individualmente

Hemos visto en el Apartado 4.1.3 cómo se inicia la actividad para mostrar un equipo individual, sin embargo, a pesar de que el título se carga en la actividad, solo disponemos de la URLs del contenido y de la Imagen.

Para poder mostrar el contenido y cargar la imagen se debe iniciar dos tipos de consultas Volley: una consulta a la página del equipo para extraer la información, y una consulta de tipo imagen para mostrar la imagen del equipo.

Ambas consultas las hemos visto anteriormente así que solamente hace falta para la primera (el contenido) parsear el resultado obtenido y extraer la información del equipo, y para la segunda añadir a la cola de peticiones una petición de tipo imagen y asociarla al contenedor de tipo NetWorkImageView de Volley.

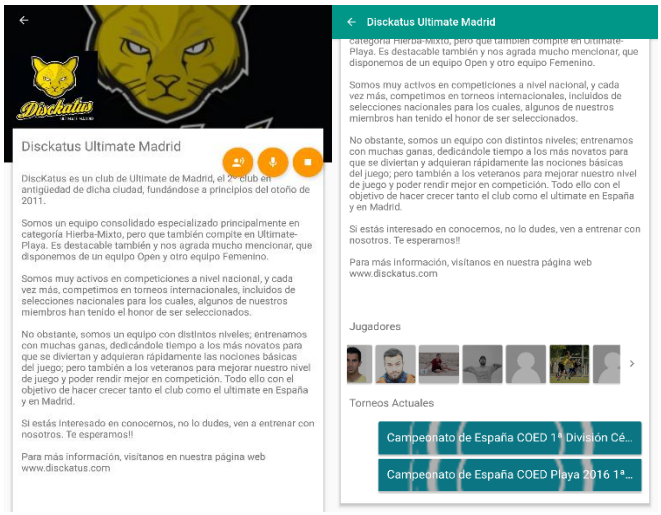


Figura 53: Ejemplo del contenido de un Ítem de tipo Equipo

4.1.6 Módulo Torneos

Vamos a ver ahora la página de torneos. Esta página muestra inicialmente los torneos que se encuentran en la ubicación que hemos introducido al registrarnos. También nos permite acceder al módulo realidad aumentada de torneos. La clase se encuentra en *páginas* y se llama *TorneoActivity.java*.

Como hemos visto anteriormente se muestra una lista con los nombres de los torneos, un breve subtítulo y la opción añadir o eliminar favoritos. La interactividad y la búsqueda son prácticamente iguales a la página de equipos, lo único que cambia es la URL de las consultas que se realizan, los adaptadores y los tipos de ítems. Aunque las clases tengan un nombre diferente estructuralmente realizan las mismas funciones.

La URL que consultan son: http://ultimatecentral.com/en_us/e/location y http://ultimatecentral.com/en_us/e/search, el adaptador se llama CardsAdaptersTorneos.java y la clase tipo ítem Item_cards_torneos.java

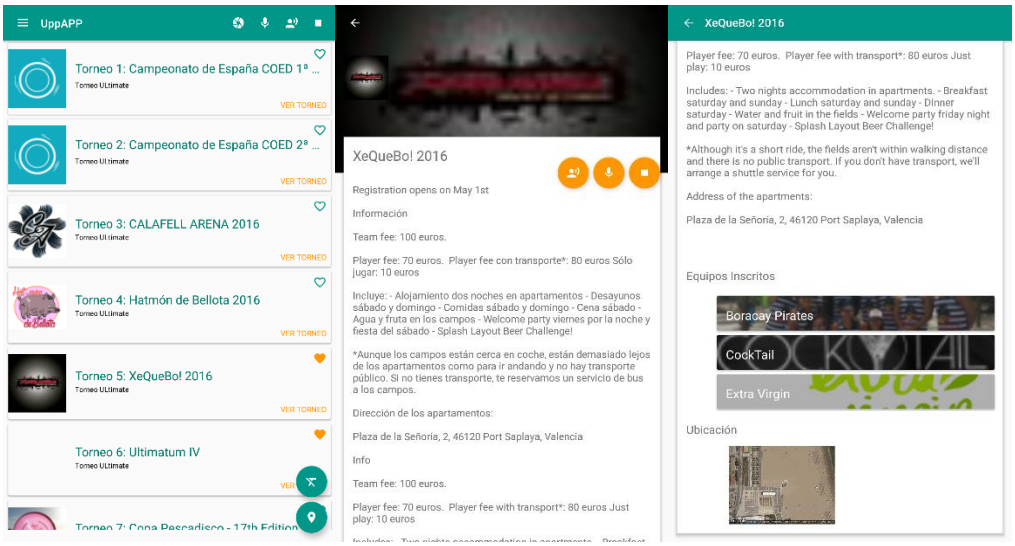


Figura 54: Captura de pantalla de la página Torneos

4.1.7 Módulo Favoritos

Favoritos es la página principal una vez se ha ingresado el usuario, es decir, es la primera página que el usuario ve cuando inicia sesión. La clase se encuentra en *páginas* y se llama *Favoritos.java.*

En esta página se pueden ver todas las reglas, torneos y equipos favoritos. Como podemos observar en la ilustración 53, los elementos están separados en 3 pestañas diferentes:

- Reglas: donde se visualizan las reglas favoritas,
- Equipos: donde se visualizan los equipos favoritos,
- Torneos: donde se visualizan los torneos favoritos. Podemos navegar por las diferentes pestañas si pulsamos en la pestaña correspondiente.

La página de favoritos, además de contener los elementos favoritos del usuario, nos permite guardar la configuración actual de nuestra aplicación. Guardar la configuración actual permite grabar en el servidor externo nuestras preferencias actuales de equipos, torneos y reglas favoritas, es decir, si hemos añadido o eliminado elementos en favoritos y se desea que estos cambios se vean reflejados en los siguientes inicios de sesión (sin importar en que dispositivo se encuentre) debemos pulsar este botón.

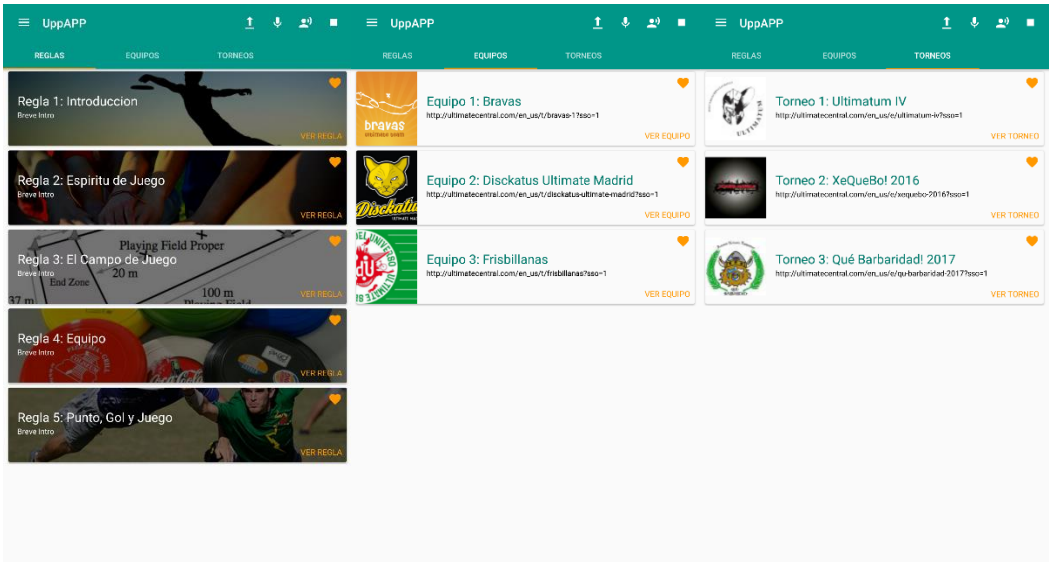


Figura 55: Captura de pantalla de la página Favoritos

<u>Carga de equipos, reglas y torneos</u>

En apartados anteriores hemos visto como se cargan los ítems dentro de cada lista: Un adaptador específico que permite la carga de los ítems, los ítems donde se guardaban los datos que hemos leído y los contenedores para mostrar dichos datos individualmente. Además hemos visto que las fuentes de datos para la reglas se encuentra en las misma aplicación, mientras que la fuente de datos de los equipos y de los torneos se encuentran fuera de la aplicación, en una página externa.

En favoritos la carga de las lista de la misma forma que en las otras páginas, es decir, crea un *RecyclerView*, *cardviews* y adaptadores para cada elemento que queremos

separar, e inicia la carga de elementos de la misma forma como se harían en sus respectivas páginas, es decir, realizamos los mismos pasos para crear las listas.

Lo único que cambia son las fuentes de datos, ya que todos los datos que queremos mostrar se encuentran en la base de datos local del dispositivo. No vamos a utilizar las referencias para mostrar el contenido en reglas, ni vamos a utilizar Volley para la carga de los datos en las listas en Torneos y equipos.

Sabemos que en el adaptador se encuentra el *array* con los ítems que van a ser mostrados, y además sabemos que podemos manipular esta lista; por ello cuando iniciamos la actividad creamos todos los ítems y los añadimos a las listas (*arrays*) de los respectivos adaptadores, notificando a los mismos que los ítems han cambiado y deben ser mostrados.

```
SQLiteDatabase db = usdbh.getWritableDatabase();
if(db != null){

    Cursor c = db.rawQuery("select * from " + UppBBDD.EquiposBBDD.TABLE_NAME,null);
    String name ="";
    String descripcion = "";//descripcion es la url de la pagina
    String jugadores = "";//no necesitamos jugadores,torneos, y pos_im ya que no se van a guardar, ya
    String torneos = "";
    String pos_img = "";
    String url_img = "";

    while(c.moveToNext()){
        name = c.getString(c.getColumnIndex(UppBBDD.EquiposBBDD.ID));
        descripcion = c.getString(c.getColumnIndex(UppBBDD.EquiposBBDD.DESCRIPCION));
        jugadores = c.getString(c.getColumnIndex(UppBBDD.EquiposBBDD.JUGADORES));
        torneos = c.getString(c.getColumnIndex(UppBBDD.EquiposBBDD.TORNEOS));
        pos_img = c.getString(c.getColumnIndex(UppBBDD.EquiposBBDD.POS_IMG));
        url_img = c.getString(c.getColumnIndex(UppBBDD.EquiposBBDD.URL_IMG));
        items_equipos.add(new Item_cards_equipos(name,descripcion,descripcion,url_img,descripcion));
        // Acciones...
    }
    db.close();
}
mAdapterEquipos.notifyDataSetChanged();
```

Figura 56: Código de la carga de ítems de tipo Equipo en favoritos

Una vez cargados los favoritos tenemos la posibilidad de eliminar ítems individualmente de las listas.

Cuando se elimina un favorito la acción siguiente debe de eliminar de la pantalla los ítems que hemos marcado como no favorito. Para que esto sea posible debemos capturar el evento *onClick* del botón favorito, para luego eliminar de la base de datos la información de dicho elemento para que la base de datos pueda estar actualizada en todo momento. Luego eliminamos de la lista de ítems el elemento marcado y notificamos al adaptador que la lista ha cambiado, por lo que en la pantalla veremos los cambios realizados.

Guardar configuración actual del sistema: reglas, torneos y equipos

Para iniciar el guardado de la configuración actual del sistema debemos capturar el evento asociado al botón  que se encuentra en la barra superior de la aplicación.

Una vez capturado el evento pasamos a iniciar la acción que nos permite guardar la información actual del usuario en la base de datos externa: **actualizar**.

Primero debemos guardar en un *array* temporal los equipos, torneos y reglas que vamos pasar al servidor. Para ello leeremos de la base de datos local los elementos favoritos del usuario y lo guardaremos los *arrays* que hemos creado. Estos *arrays* nos servirán más adelante para enviar los datos en el mensaje de consulta.

Las peticiones se van a realizar al servidor externo por lo que utilizaremos de nuevo Retrofit para realizar las llamadas y obtener las respuestas.

Como hemos visto Retrofit necesita una interfaz para saber dónde se encuentra el archivo por el que preguntará, por ello vamos a reutilizar *RequestInterface.java* para realizar la consulta y pasárselo al constructor de la clase.

Por otra parte, cuando hacemos la llamada el servidor debe saber que vamos a insertar las reglas, equipos y torneos de un usuario. Para ello contamos con la clase *user* que hemos visto antes y que guardará en arrays los equipos, reglas y torneos favoritos (si es que existen favoritos). Los arrays que contiene *users* son los que hemos guardado antes al leer la base de datos local, y que pasaremos como referencia para que esta clase pueda manejarlos.

La clase *user* se encapsula en el mensaje de consulta enviado junto con el tipo de mensaje y la operación a realizar: insertar reglas, insertar equipos e insertar torneos.

El servidor recibe los mensajes y comprueba que el usuario existe. Si existe se borran de las tablas de equipos, reglas y torneos favoritos la información relativa al usuario, e inserta en esas mismas tablas la nueva información que ha venido por mensaje.

Si hay éxito el servidor devuelve un mensaje de éxito a la aplicación, si no es así el servidor, devuelve un mensaje de error.

```
RequestInterface requestInterface = retrofit.create(RequestInterface.class);
User user = new User();
user.setUnique_id(unique_id);
user.setReglas(regla);
ServerRequest request = new ServerRequest();
request.setOperation(Constantes.INSERTAR_REGLAS);
request.setUser(user);

Call<ServerResponse> response = requestInterface.operation(request);

response.enqueue(new Callback<ServerResponse>() {
    @Override
    public void onResponse(Call<ServerResponse> call, retrofit2.Response<ServerResponse> response) {

        ServerResponse resp = response.body();

        if(resp.getResult().equals("Exito")){
            Toast.makeText(getApplicationContext(), "Actualizada tabla reglas", Toast.LENGTH_SHORT).s

            //si la respuesta es afirmativa entonces decimos qie se ha actualizado
        }
        else{
            Toast.makeText(getApplicationContext(), "Listo", Toast.LENGTH_SHORT).show();
        }
```

Figura 57: Ejemplo de inserción de las reglas en la base de datos externa

4.1.8 Módulo Perfil

En perfil podemos ver la información relativa al usuario, como el nombre, la dirección de correo y la ciudad elegida.

Por otra parte, permite cambiar la ciudad elegida y la contraseña. En ambos casos hace falta una verificación de la contraseña para realizar el cambio. La clase se encuentra en *páginas* y se llama *setings.java.*

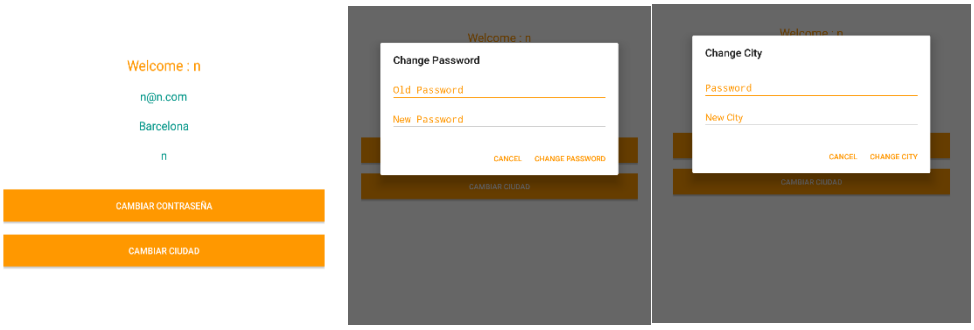


Figura 58: Captura de pantalla de la página Perfil

Para poder realizar los cambios debemos comunicarnos con el servidor externo, por lo que utilizaremos Retrofit y las clases que hemos creado anteriormente serán reutilizadas.

Esta vez el tipo de operación es: cambiar ciudad o cambiar contraseña. Cuando se envía el mensaje de con la consulta enviamos en la clase *user* la contraseña del usuario con la contraseña nueva, en caso de cambiar la contraseña, y la contraseña con la ciudad en caso que se quiera cambiar la ciudad.

El servidor recibe la consulta, comprueba el usuario, verifica la contraseña, y si todo es correcto realiza los cambios en las base de datos de los usuarios, para luego enviar un mensaje de éxito a la aplicación. Si ha ocurrido un error de verificación el servidor envía un mensaje de este tipo. Si existe un error en la comunicación Retrofit captura el error y lo notifica al usuario.

4.1.9 Módulo Información

Está página muestra a información propia de la aplicación. La clase se encuentra en *páginas* y se llama *informacion.java*.

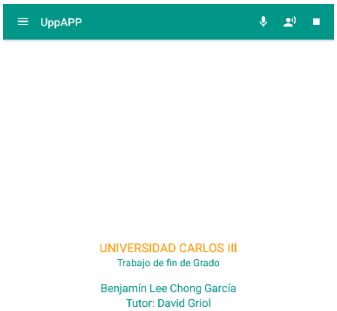


Figura 59: Captura de pantalla de la página Información

4.2 Módulo asistente oral.

El módulo *asistente oral* nos permite por una parte acceder mediante la voz a las distintas funcionalidades de la aplicación, y por otra escuchar las instrucciones y el contenido de las páginas en las que se encuentra.

Debido a la naturaleza propia del módulo, su implantación se realiza a lo largo de todo el proyecto, creando en cada actividad las clases y funciones necesarias para que pueda ser utilizado.

Para la lectura de las instrucciones y del contenido utilizaremos la síntesis de voz como salida, y como entrada utilizaremos el reconocimiento de voz.

El módulo *asistente* sonoro además realiza todas las acciones que normalmente se harían por medio del teclado táctil, como: ir a inicio, registrarse, ir a favoritos, reglas, equipos, información o perfil, salir de la aplicación, búsqueda de torneos y equipos, búsqueda de palabras y frases en reglas, añadir y eliminar favoritos, guardar la configuración del equipo y empezar la actividad de realidad aumentada tanto en equipos como en torneos.

Este módulo se encarga en resumen de:

- Informar al usuario de las opciones que este tiene cuando se inicia el módulo.
- Capturar la voz de entrada con las indicaciones del usuario, y realizar la acción descrita.
- Leer contenido de una página en concreto.

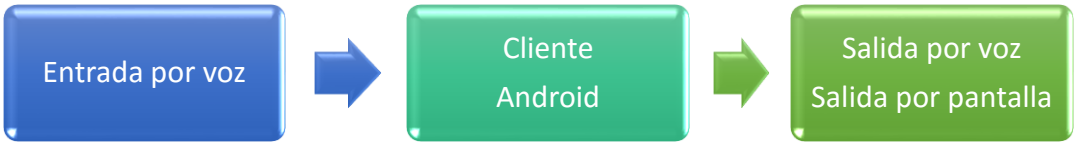


Figura 60: Estructura del módulo oral

Para poder iniciar el módulo hemos provisto a la aplicación por una parte de un botón de síntesis de voz que nos permite escuchar las instrucciones que debemos seguir para iniciar la acción, otro botón para reconocer la voz del usuario y por último un botón que nos permite parar la síntesis de voz. Estos botones se encuentran normalmente en la barra superior de la aplicación, situados a la derecha de la misma.

Por otra parte, para mejorar la interactividad con el dispositivo móvil, debemos activar el asistente de síntesis de voz de Android para saber en cualquier momento que objeto de la pantalla estamos tocando.

Hemos visto en los Apartados 2.7.3 y 2.7.4 qué clases e importaciones debemos hacer a nuestras actividades para poder hacer funcionar el sistema de diálogo, por lo que en este apartado nos centraremos en el flujo de información entre el asistente sonoro y la aplicación.

4.2.1 Escuchar información e instrucciones de la página

Este apartado se expondrá la funcionalidad del botón escuchar información.

En primer lugar deben estar declarados los objetos necesarios para la síntesis de voz:

- Debemos importar la clase *TextToSpeech* y crearla dentro de la clase principal.
- Cuando inicializamos la clase principal añadimos a la clase *TextToSpeech* un *listener* que permite inicializar la síntesis de voz, cuando se inicializa la síntesis de voz se inicia un mensaje de bienvenida propio de la página. Este mensaje nos indica donde nos encontramos y nos dice que existen los botones de escucha y reconocimiento de habla.
- Dentro de la inicialización debemos comprobar que los lenguajes son soportados y las versiones son soportadas
- Una vez realizado esta comprobación el mensaje de bienvenida será pasado a la función *speak* (Apartado 2.7.4) que iniciará la lectura del mensaje por la librería *Android.Speech*.

```
TextToSpeech.OnInitListener listener = new TextToSpeech.OnInitListener() {
    @Override
    public void onInit(int status) {
        if (status == TextToSpeech.SUCCESS) {

            Locale spanish = new Locale("es", "ES");
            int result = tts.setLanguage(spanish);
            if (result == TextToSpeech.LANG_MISSING_DATA || result == TextToSpeech.LANG_NOT_SUPPORTED) {
                Log.e("TTS", "This Language is not supported");
            }

            if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP) {
                String bienvenida = getResources().getString(R.string.MensajeBienvenida_login);
                tts.speak(bienvenida , TextToSpeech.QUEUE_FLUSH, null, null);

            }else{
                String bienvenida = getResources().getString(R.string.MensajeBienvenida_login);
                tts.speak(bienvenida, TextToSpeech.QUEUE_FLUSH, null);
                tts.speak(bienvenida, TextToSpeech.QUEUE_FLUSH, null);
            }
```

Figura 61: Inicialización de la librería Android.Speech

Una vez inicializada la librería, vamos a pasar a capturar el evento relacionado con el botón *escucha de información*. Este botón se encuentra en la barra superior por lo que podemos capturar el evento gracias la función onClick de la misma. De esta forma cuando el botón es pulsado se inicia la síntesis de las instrucciones asociadas a la página donde nos encontremos. Las instrucciones se encuentran en el archivo *Strings* de la subcarpeta *values* de la carpeta *res*.

```
case R.id.img_btn_escuchar_login:
    tts.stop();
    //escucha_info.setBackgroundResource(R.drawable.voice_anunciandose);
    speak(getResources().getString(R.string.instrucciones_login));
```

Figura 62: Inicio de la síntesis de instrucciones en la página de Login

Podemos observar que antes de utilizar la función *speak* paramos todas las síntesis que se están ejecutando, ya que de esta forma será la lectura de las instrucciones se leerá inmediatamente. *Tts.stop()* nos será útil para parar las síntesis actuales y empezar unas nuevas a lo largo de la aplicación. Así podemos organizar las síntesis sin que haya retrasos o la síntesis actual tenga que esperar para reproducirse.

4.2.2 Reconocimiento de indicaciones por voz

Pasamos a capturar el reconocimiento de voz cuando se pulsa el botón micrófono de la aplicación  .

Como pasaba con la síntesis de voz debemos crear y declarar las clases que nos hacen falta y que vienen descritas en el Apartado 2.7.3 del presente documento:

- En primer lugar importamos y creamos el *RecognizerIntent* dentro de la clase principal.
- Debemos capturar ahora el evento *onClick* del botón, y crear el *itent* que proporciona la funcionalidad para empezar la actividad de reconocer la voz.
- Añadimos los extras, tales como el lenguaje que va a reconocer, en este caso, el español, aunque puede ser un lenguaje libre.
- Iniciamos la actividad que recogerá la voz del usuario.

```
case R.id.img_btn_hablar_login:
    tts.stop();
    Intent intent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
    intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL, "es");
    intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL, RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);
    try {
        startActivityForResult(intent, RECOGNIZE_SPEECH_ACTIVITY);
    } catch (ActivityNotFoundException a) {
        Toast.makeText(getActivity().getApplicationContext(), "Tu dispositivo no soporta el reconocimie
    }
    break;
```

Figura 63: Ejemplo de la inicialización de la captura de voz.

Una vez finalizado la captura de voz la aplicación devolverá el resultado de la misma por medio de la función *onActivityResult*. Es en este método donde se devolverá en forma de texto la información introducida por el usuario, y estará presente en todas las actividades que queremos que puedan capturar las indicaciones del usuario.

```
public void onActivityResult(int requestcode, int resultcode, Intent datos)
{
    if (resultcode== Activity.RESULT_OK && datos!=null)
    {
        ArrayList<String> text=datos.getStringArrayListExtra(RecognizerIntent.EXTRA_RESULTS);

        if(text.get(0).toLowerCase().equals("aceptar")){
                btn_login.callOnClick();
        }
        if(text.get(0).toLowerCase().equals("ir a registro")){
                goToRegister();
```

Figura 64: Ejemplo de reconocimiento de voz y conversión a texto en la página de Login

En la Figura anterior podemos observar la captura de voz en la página de inicio de sesión. El texto viene dado por *arrayList* de tipo *String* y que contiene los datos de la consulta.

Vemos además que para comparar los resultados utilizamos la primera posición del texto (*text.get(0)*) y lo convertimos a minúsculas para poder realizar la comparación.

En este ejemplo vemos que si la consulta coincide puede ir a la página de registro, o iniciar el inicio de sesión del usuario.

Esta estructura se utilizará de forma más o menos desarrollada a lo largo de la aplicación dependiendo las opciones que se tengan en la página donde se encuentren, por ejemplo, en el ejemplo solo utilizamos la primera posición del array, sin embargo, en otras páginas utilizaremos más posiciones para verificar el contenido de la consulta.

A continuación se describe brevemente las acciones que se pueden realizar con el reconocimiento de voz en las diferentes páginas. Las acciones que se presentan deben llamarse tal cual están escritas en esta sección:

Acción	**Paginas donde se encuentra**	**Descripción**
registrar	RegitroFragment.java	Empieza el registro de un nuevo usuario
ir a principal	RegitroFragment.java	Inicia la actividad inicio de sesión.
aceptar	LoginFragment.java	Empieza el inicio de sesión del usuario.
ir a registro	LoginFragment.java	Inicia la actividad Registro.
ir a (Favoritos, equipos, torneos, reglas, perfil, información)	EquipoActivity.java Favoritos.java informacion.java ReglasActivity.java setings.java TorneoActivity.java	Cambia la actividad a algunas de las actividades siguientes: Favoritos, equipos, torneos, reglas, perfil e información; excepto si se encuentra en la misma actividad.
Ir a salir	EquipoActivity.java Favoritos.java informacion.java ReglasActivity.java setings.java TorneoActivity.java	Sale de la aplicación y vuelve a la actividad LoginFragment.java
oír equipos	EquipoActivity.java Favoritos.java punto_de_interes_Equipos.java	Llama a la síntesis de voz y lee los equipos que se encuentran cargados en la pantalla actual.
equipo “número de equipo”	EquipoActivity.java punto_de_interes_Equipos.java	Selecciona de forma individual un equipo que se encuentra en la lista, e inicia la actividad para ver su contenido si se encuentra en EquipoActivity.java, o muestra la información referente a su posición si se encuentra en punto_de_interes_Equipos.java. Si el

Acción	Paginas donde se encuentra	Descripción
		equipo no existe se escucha un mensaje erróneo.
equipo “número de equipo” favorito	EquipoActivity.java	Añade el equipo seleccionado como favorito. Si el equipo no existe se escucha un mensaje erróneo. Si existe y es favorito la aplicación informa al usuario de este suceso.
equipo “número de equipo” no favorito	EquipoActivity.java Favoritos.java	Añade el equipo seleccionado como favorito. Si el equipo no existe se escucha un mensaje erróneo. Si existe y no es favorito la aplicación informa de este suceso.
buscar zona “zona”	EquipoActivity.java TorneoActivity.java	Una vez dicha la palabra, inicia la búsqueda de equipos o torneos por zona, y como zona utiliza la palabra introducida por el usuario.
buscar equipo “equipo”	EquipoActivity.java	Una vez dicha la palabra, inicia la búsqueda de equipos por nombre, y como nombre utiliza la palabra introducida por el usuario.
oír torneos	TorneoActivity.java Favoritos.java punto_de_interes_Torneos.java	Llama a la síntesis de voz y lee los torneos que se encuentran cargados en la pantalla actual.
Torneo “número de torneo”	TorneoActivity.java punto_de_interes_Torneos.java	Selecciona de forma individual un torneo que se encuentra en la lista, e inicia la actividad para ver su contenido si se encuentra en TorneoActivity.java, o muestra la información referente a su posición si se encuentra en punto_de_interes_Torneos.java. Si el torneo no existe se escucha un mensaje erróneo.
torneo “número de torneo” favorito	TorneoActivity.java	Añade el torneo seleccionado como favorito. Si el torneo no existe se escucha un mensaje erróneo. Si existe y es favorito la aplicación informa al usuario de este suceso.
torneo “número de torneo” no favorito	TorneoActivity.java Favoritos.java	Añade el torneo seleccionado como favorito. Si el torneo no existe se escucha un mensaje erróneo. Si existe y no es favorito la aplicación informa de este suceso.
buscar torneo “torneo”	TorneoActivity.java	Una vez dicha la palabra, inicia la búsqueda de torneos por nombre, y como nombre utiliza la palabra introducida por el usuario.

Acción	Paginas donde se encuentra	Descripción
oír reglas	ReglasActivity.java Favoritos.java	Llama a la síntesis de voz y lee las reglas que se encuentran cargados en la pantalla actual.
cargar reglas	ReglasActivity.java	Cargas todas las reglas.
regla “número de regla”	ReglasActivity.java	Selecciona de forma individual una regla que se encuentra en la lista, e inicia la actividad para ver su contenido. Si la regla no existe se escucha un mensaje erróneo.
regla “número de regla” favorito	ReglasActivity.java	Añade la regla seleccionada como favorito. Si la regla no existe se escucha un mensaje erróneo. Si existe y es favorito la aplicación informa al usuario de este suceso.
regla “número de regla” no favorito	ReglasActivity.java Favoritos.java	Añade la regla seleccionada como favorito. Si la regla no existe se escucha un mensaje erróneo. Si existe y no es favorito la aplicación informa de este suceso.
buscar “frase”	ReglasActivity.java	Una vez dicha la palabra o palabras, inicia la búsqueda de términos, palabras y frases de la página ReglasActivity.java.
Leer	TorneoContenidoActivity.java ReglaContenidoActivity.java EquipoContenidoActivity.java informacion.java	Lee el contenido de la actividad actual, es decir, lee texto que se encuentra dentro de la página.
subir	Favoritos.java	Permite iniciar la acción guardar la configuración actual del usuario de la página de Favoritos.java
cambiar contraseña	setings.java	Inicia la acción de cambiar contraseña de la actividad setings.java
cambiar ciudad	setings.java	Inicia la acción de cambiar ciudad de la actividad setings.java
volver	TorneoContenidoActivity.java ReglaContenidoActivity.java EquipoContenidoActivity.java punto_de_interes_Torneos.java punto_de_interes_equipo.java MapsActivity.java	Permite volver a la actividad anterior en la que se encontraba la aplicación.
Realidad aumentada	EquipoActivity.java TorneoActivity.java	Inicia la actividad de realidad aumentada.
Torneo “número de torneo” ruta	punto_de_interes_Torneos.java	Inicia la actividad mostrar mapa con ruta del torneo seleccionado.
Equipo “número de torneo” ruta	punto_de_interes_Equipos.java	Inicia la actividad mostrar mapa con ruta del equipo seleccionado.

Acción	Paginas donde se encuentra	Descripción
Iniciar ruta	MapsActivity.java	Inicia la aplicación Google Maps con la ruta hacia el torneo o equipo seleccionado.

Tabla 10: Mandatos de voz que reconoce la aplicación

Todas las acciones tienen como resultado una salida de voz, ya sea de confirmación o de lectura del contenido.

Un detalle de diseño es la forma en cómo seleccionamos por voz las reglas, equipos y torneos. En todos los casos asignamos un número a los ítems que van apareciendo en la pantalla. Este número coincide con la posición siguiente del ítem dentro la lista, es decir, si el ítem se encuentra en la posición cero se mostrará en la pantalla para este ítem el texto "uno" seguido del título del equipo o torneo. Por ello cuando seleccionamos un ítem por voz empezamos diciendo el tipo de ítem (equipo, torneo o regla) seguido del número que lo identifica.

Hemos implementado este diseño ya que: los títulos de las reglas varían de idioma, es decir, un título puede tener palabras en español o en inglés. Con respecto a los equipos y torneos, además de tener el mismo problema, surge esta necesidad ya que varios títulos de equipos se repiten, por lo que hay que diferenciar los unos de los otros.

4.3 Módulo Realidad Aumentada

El módulo *Realidad Aumentada* nos permite acceder a las funciones de búsqueda de equipos y torneos por medio de la geolocalización en tiempo real, utilizando la cámara del dispositivo. Además añade la funcionalidad de poder acceder a la ruta entre el usuario y el equipo o torneo seleccionado.

Este módulo utiliza dos tipos de tecnologías para poder representar el entorno de realidad aumentada: nativa (Java), web (JavaScript, HTML, JQuery y CSS). Wikitude separa los espacios de desarrollo de estas tecnologías, situando ambas en carpetas diferentes, y las comunica cuando se ejecuta el RA.

Hemos visto en el Apartado 2.8.3 que para la visualización del entorno RA hace falta: el código nativo Java para gestionar la actividad e inicializar el motor RA, y la tecnología web que permite inicializar y capturar los elementos y eventos que se mostrarán en la RA.

Para cargar y crear los elementos que va a conformar el espacio en la RA se debe utilizar el *script* de la aplicación. Sin embargo, para obtener los datos que permitirán la creación de esos elementos debemos realizar una consulta al servidor externo.

Este servidor recibirá una consulta del JavaScript utilizando el método ***GetJson()***, y devolverá los elementos (torneos o equipos) en formato JSON para su posterior conversión y creación por el archivo JavaScript.

Por otra parte, hemos visto que JavaScript permite comunicar eventos al código nativo por medio de la llamada de tipo *URL Listener*, de esta forma cuando ocurre un evento que nos interesa podemos manejarlo.

Cabe recalcar que es nuestro servidor externo, no la página web, a la que se hacen las peticiones, ya que la web no cuenta con una API desarrollada para estos casos.

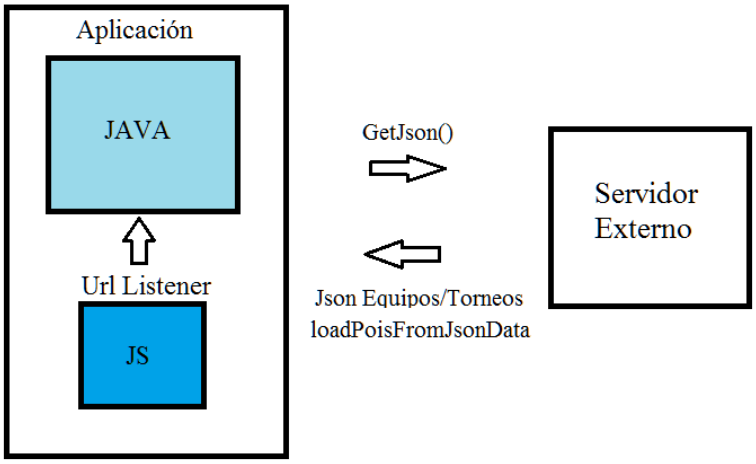


Figura 65: Comunicación del Módulo Realidad Aumentada con el servidor externo

4.3.1 Módulo Java

Las acciones principales de esta que debe realizar esta tecnología deben ser:

- Crear la actividad donde se contendrá la interfaz RA.
- Crear y gestionar los estados que aparecen en el ciclo de vida de la actividad: onCreate(), onPostCreate(), etc.
- Inicializar el motor RA.
- Capturar los eventos que provienen de la parte web de la aplicación (*url Listener*).

En el Apartado 2.8.3 podemos ver las clases Java que necesitamos para poder empezar a utilizar el AR: *AbstractArchitectCamActivity.java; ArchitectViewHolderInterface.java y LocationProvider.java;* todas ellas se encuentran en la carpeta *Realidad Aumentada.*

Tenemos además en esta carpeta dos clases más *punto_de_interes_Equipos.java* y *punto_de_interes_Torneos.java* Estas dos clases serán las actividades principales que contendrán las interfaces AR para equipos y para torneos.

Estas actividades serán llamadas desde la página de equipos o de la página de torneos, abriéndose la clase *punto_de_interes_Torneos.java* para equipos, y *punto_de_interes_Torneos.java* para torneos.

Las clases punto de interés serán las clases que extiendan de *AbstractArchitectCamActivity.java*: clase abstracta que permite la carga de la interfaz RA

y la gestión de los estados necesarios para poder controlar el ciclo de vida de nuestra actividad. Cuando se inicia recibe el XML que corresponde a la interfaz de la actividad, y los sensores que debe iniciar. En esta clase también se inicializa la síntesis de audio comentado en el módulo anterior y recibe la clave Wikitude para la utilización del sistema en la aplicación.

En el ciclo de vida de la aplicación suceden los siguientes estados:

- OnCreate: inicialización del RA. Paso de la clave Wikitude para el acceso a la aplicación. Se crea el listener para capturar la localización actual y para capturar el cambio de posición. Se inicializa el sintetizador de voz (módulo asistente sonoro).

- OnPostCreate: Carga el sistema RA una vez hemos finalizado el estado OnCreate.

- OnResume: Se inicia la cámara del dispositivo, y se superpone la interfaz web. En la parte web se empieza a cargar los elementos que se van a representar en la cámara, por lo que empieza la comunicar con el servidor externo. Además OnResume captura los eventos sucedidos en la parte web, para ello utilizamos el urlListener

- OnLowMemory: Informa si el dispositivo no tiene suficiente memoria para trabajar.

- OnDestroy: Libera todos los recursos utilizados por la actividad, y la finaliza.

Las actividades que heredan de *AbstractArchitectCamActivity.jav*a son entonces las que pueden iniciar el sistema RA y por consiguiente son las que deben pasar que tipos de sensores se van a activar cuando se inicia el motor RA.

Los sensores de localización se activarán y gracias a la clase *ILocationProvider.java* podremos pasar y saber nuestra posición actual. Además añadimos al sensor una calibración para que los datos se muestran más fijos en la pantalla.

Por otra parte *punto_de_interes_Equipos.java* y *punto_de_interes_Torneos.java* capturan el *listener* para los eventos surgidos en la parte web. Para ello sobrescribimos el método que captura el *listener* y configuramos las acciones a realizar.

En nuestro caso los eventos que se van a capturar son:

- *Markerselected:* si un ítem es seleccionado se iniciara la actividad correspondiente a ese ítem, mostrando todo la información de ese equipo o torneo en una pantalla diferente.

- *verMapaElemento:* muestra en una actividad diferente la ruta que existe entre el ítem seleccionado y el usuario.

- *darSeleccion:* invoca el *RecognizerIntent* para capturar la voz del usuario, realizando las acciones descritas en el módulo asistente sonoro.

```
@Override
public ArchitectView.ArchitectUrlListener getUrlListener() {
    return new ArchitectView.ArchitectUrlListener() {

        @Override
        public boolean urlWasInvoked(String uriString) {

            Uri invokedUri = Uri.parse(uriString);
            tts.stop();
            // pressed "More" button on POI-detail panel
            if ("markerselected".equalsIgnoreCase(invokedUri.getHost())) {
                final Intent poiDetailIntent = new Intent(punto_de_interes_Equipos.this, EquipoContenidoActivity.class);
                //poiDetailIntent.putExtra("ID", String.valueOf(invokedUri.getQueryParameter("id")) );
                poiDetailIntent.putExtra("NOMBRE", String.valueOf(invokedUri.getQueryParameter("title")) );
                poiDetailIntent.putExtra("URL", String.valueOf(invokedUri.getQueryParameter("description")) );
                poiDetailIntent.putExtra("URLIMAGEN", String.valueOf(invokedUri.getQueryParameter("url_imagen")) );
                punto_de_interes_Equipos.this.startActivity(poiDetailIntent);
                return true;
```

Figura 66: Ejemplo de captura urlListener cuando un elemento ha sido seleccionado

Todos los eventos que capturamos tienen como parámetro de entrada una URL que contiene los datos que necesitamos para realizar la acción.

Los datos tenemos que transfórmalos para poder utilizarlos. Por ello utilizamos un objeto de tipo URI para parsear su contenido y así poder acceder de forma más sencilla.

4.3.2 Módulo JavaScript y HTML

Cuando ya hemos inicializado la interfaz RA, debemos cargar los elementos que componen la vista del mismo. Para ello, cargamos los elementos comunes que no necesitan comunicación con otras partes de la aplicación, y utilizamos las librerías que Wikitude nos deja a disposición.

De esta forma, podemos crear los botones para la escucha y para la captura de voz, podemos crear el radar que nos indica los puntos de interés que se encuentran cargado y en qué dirección se encuentran, o podemos crear el menú superior para las demás opciones que queramos poner al sistema.

En los anteriores apartados vimos que a la actividad RA se le superpone una página HTML para la visualización de los elementos. La página HTML carga todas las librerías JQuery, CSS y JS necesarias para la visualización de los paneles y elementos que conforman la vista de la cámara.

La implementación de la experiencia RA se llama *World* y es aquí donde se guardan todos los elementos del RA.

En *World* se crean los elementos que no necesitan una interactividad con la parte nativa, creando los paneles, elementos y *divs* necesarios en la página HTML. Los elementos o paneles pueden llevar una interactividad por lo que capturamos los eventos que necesitamos y generamos las animaciones y las acciones que se van realizar, por ejemplo, abrir los detalles.

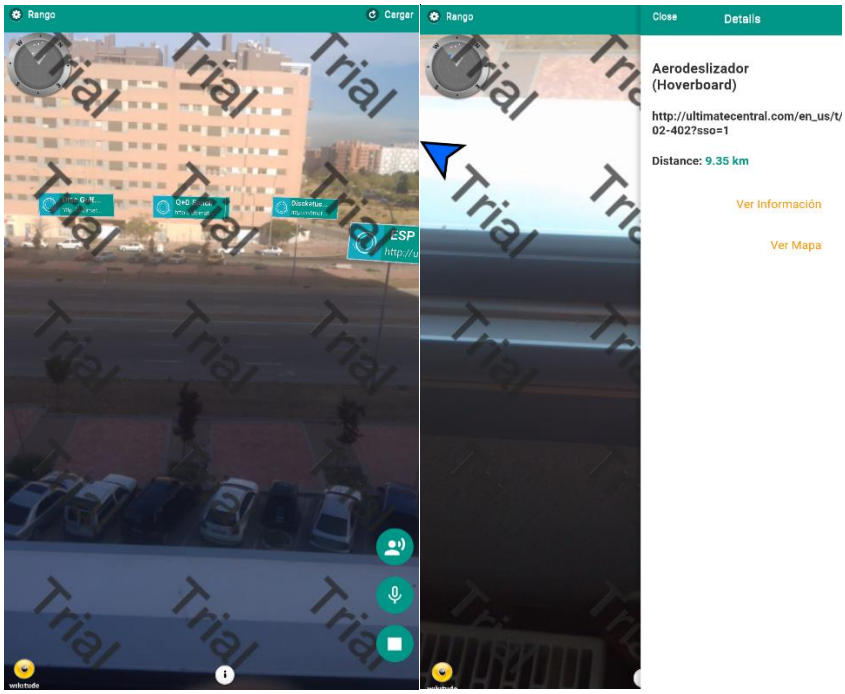


Figura 67: Captura de pantalla de la página Realidad Aumentada de Equipos

La captura de todos los eventos se realizan en los archivos JavaScript asociados a las páginas correspondientes.

Por último, nos falta cargar los puntos de interés que se van a mostrar. Estos puntos son creados a partir de una clase llamada *markers.js* cuyos datos son extraídos a través de la consulta que hacemos al servidor.

La consulta se hace en los JavaScript principales cuyo método a invocar es *requestDataFromServer*. Este método crea una petición GET que envía al servidor externo, devolviendo este último, si todo ha salido correctamente, un objeto en formato JSON con la información relativa a los torneos o equipos. El archivo JavaScript interpretará el objeto e irá creando los objetos de tipo marker gracias al método *loadPoisFromJsonData*.

```
loadPoisFromJsonData: function loadPoisFromJsonDataFn(poiData) {
    AR.context.destroyAll();
    PoiRadar.show();
    $('#radarContainer').unbind('click');
    $("#radarContainer").click(PoiRadar.clickedRadar);
    World.markerList = [];
    World.markerDrawable_idle = new AR.ImageResource("assets/marker_idle.png");
    World.markerDrawable_selected = new AR.ImageResource("assets/marker_selected.png");
    World.markerDrawable_directionIndicator = new AR.ImageResource("assets/indi.png");
    for (var currentPlaceNr = 0; currentPlaceNr < poiData.length; currentPlaceNr++) {
        var singlePoi = {
            "id": poiData[currentPlaceNr].id,
            "latitude": parseFloat(poiData[currentPlaceNr].latitude),
            "longitude": parseFloat(poiData[currentPlaceNr].longitude),
            "altitude": parseFloat(poiData[currentPlaceNr].altitude),
            "title": poiData[currentPlaceNr].name,
            "description": poiData[currentPlaceNr].description,
            "url_imagen": poiData [currentPlaceNr].url_img
        };
        World.markerList.push(new Marker(singlePoi));
    }
    World.updateDistanceToUserValues();
    World.updateStatusMessage(currentPlaceNr + ' places loaded');
    $("#panel-distance-range").val(100);
    $("#panel-distance-range").slider("refresh");
```

Figura 68: Carga de los elementos.

Como hemos dicho antes el archivo JavaScript debe implementar todas los eventos ocurridos en la RA, y en algunos casos debe pasar el evento a la parte Nativa. Para pasar los eventos a la parte nativa utilizamos la función utilizamos la función *document.location*. Esta función nos permite pasar los datos en forma de URL con un método POST.

```
var architectSdkUrl = "architectsdk://darSeleccion?torneos=" + torneos
                    + "&distancias=" + distancias;
document.location = architectSdkUrl;
```

Figura 69: Ejemplo de invocación a Java desde JavaScript

- onPoiDetailMoreButtonClicked: selección de un punto de interés. Abre un panel lateral donde se muestra los detalles de un equipo o un torneo. El panel lateral tiene además dos botones que permiten ir al contenido del elemento seleccionado, o mostrar su ruta.
- onMapButtonClicked: actividad que se inicia al apretar el botón *ver mapa*. Muestra la ruta existente entre el usuario y el equipo o torneo seleccionado. Esta actividad debe pasar a la actividad nativa el evento para que esta pueda iniciar la actividad correspondiente, además de los datos de localización.
- escucharInstruccionesClickedFn: Se comunica con la parte nativa para iniciar la síntesis de voz de las instrucciones de la página.
- darSeleccionClickedFn: Se comunica con la parte nativa para iniciar el reconocimiento de voz con la página. Pasa los equipo o torneos que hay cargados. De esta forma podemos comprobar si el equipo nombrado existe y si se puede seleccionar.
- darTorneoAConsultar, darEquipoAConsultar: estas funciones se comunican con la parte nativa para iniciar la actividad que muestra el contenido del equipo o torneo seleccionado.

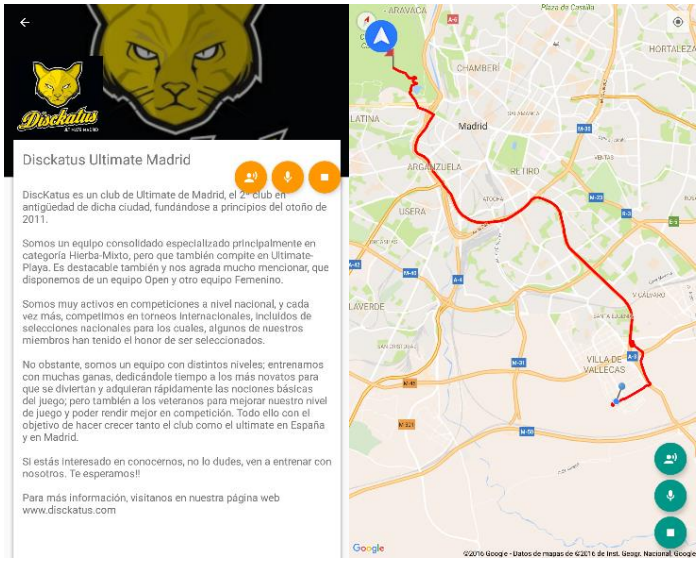


Figura 70: Ejemplo con el contenido mostrado en RA y en la actividad Ruta

Por último, vamos a ver cuáles son los archivos HTML que cargan las librerías y que arvhivos JS cargan y contienen los *Worlds: index_equipos.html* e *index_torneos.html* son los archivos que se comunican con *punto_de_interes_Equipos.java* y *punto_de_interes_Torneos.java* respectivamente.

Los archivos JS que contienen los *worlds* y sus funciones son: *nativedetailscreenTorneos.js y nativedetailscreenEquipos.js*. Estos archivos contienen la información del servidor para realizar la consulta, en el caso de equipos es ***http://localhost/api.upp/Equipos*** y el de torneos es ***http://localhost/api.upp/Torneos***, y contienen el rango y las funciones necesarias para asignar el rango de visibilidad de los elementos y escoger los elementos que se van a mostrar.

```
requestDataFromServer: function requestDataFromServerFn(lat, lon) {
    World.isRequestingData = true;
    World.updateStatusMessage('Requesting places from web-service');
    var serverUrl = ServerInformation.POIDATA_SERVER + "?"
                        + ServerInformation.POIDATA_SERVER_ARG_LAT
                        + "=" + lat + "&" + ServerInformation.POIDATA_SERVER_ARG_LON
                        + "=" + lon + "&"
                        + ServerInformation.POIDATA_SERVER_ARG_NR_POIS + "=20";
    var jqxhr = $.getJSON(serverUrl, function(data) {
            World.loadPoisFromJsonData(data);
        })
        .error(function(err) {
            World.updateStatusMessage("Invalid web-service response.", true);
            World.isRequestingData = false;
        })
        .complete(function() {
            World.isRequestingData = false;
```

Figura 71: Código ejemplo del Request Server desde JS

4.3.3 Obtener Ruta

En el apartado anterior se describía una de las funciones que tiene el módulo RA: Una vez seleccionado un elemento es posible mostrar la ruta a este elemento desde la posición del usuario. Hemos visto además, que es esto se hace pasando por varias fases: primera la selección del elemento por medio de funciones JavaScrit, la captura del evento cuando el botón es pulsado, y el paso del evento y datos a la parte nativa de la aplicación.

El botón que se pulsa es *Ver Mapa*, y los datos que se pasan son los correspondientes al nombre y la posición, en términos de longitud y latitud, del elemento seleccionado.

Cuando tenemos ambos datos la aplicación inicia la actividad *MapsActivity.java* que se encuentra en la carpeta *páginas*, e inicia el cargado del mapa y la ruta gracias a la librería Maps de Google.

Cuando tenemos el mapa cargado y lo hemos iniciado, debemos cargar los *markers* que sitúan los puntos donde se encuentran, en primer lugar el usuario, y en segundo lugar el elemento seleccionado. Para ello utilizamos el método de la librería ***onLocationChanged***. Este método carga los marcadores y los coloca en el mapa. Una vez cargados calculamos la ruta gracias al método ***FetchUrl.execute***.

```
String url = getUrl(mCurrLocationMarker.getPosition(), destino);
FetchUrl FetchUrl = new FetchUrl();
FetchUrl.execute(url);
```

Figura 72: Ejemplo de inicialización de una ruta en la Actividad Ruta

Por último, tenemos la posibilidad de escuchar las instrucciones de la ruta apretando el botón, el cual iniciará la aplicación *Google maps* de Android.

4.4 Flujo de datos

En este apartado veremos el flujo de datos que existe entre las distintas páginas, y las acciones que podemos realizar para cada una de ellas.

Logo inicial

El logo de la aplicación será el siguiente, y representa a un jugador de lanzándose hacia un Frisbee.

Figura 73: Logo de la aplicación

Inicio de Sesión.

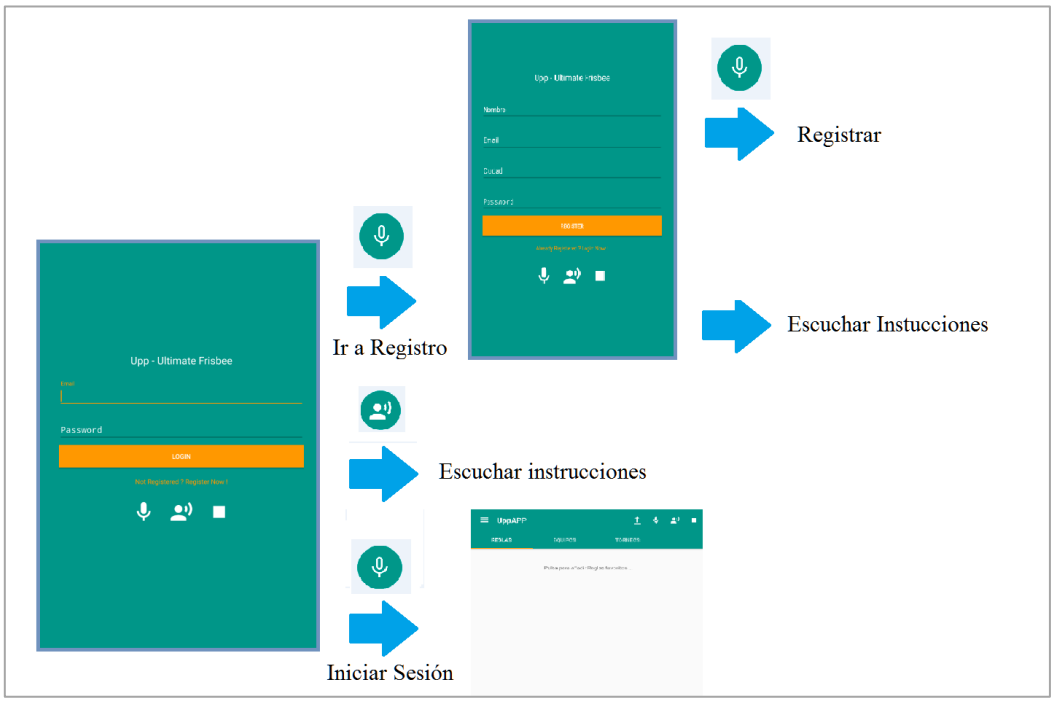


Figura 74: Flujo desde la página Inicio de sesión

- De página Inicio de sesión podemos ir a la página Registro al presionar en *Register now* o por medio del comando de voz *ir a registro.*
- En la página Inicio de sesión podemos escuchar las instrucciones de la página si presionamos el botón
- De la página Inicio de sesión podemos ir a la página Favoritos al presionar en *Login* o por medio del comando de voz *aceptar.*
- De página Registro podemos ir a la página Inicio de sesión al presionar en *Login now* o por medio del comando de voz *ir a principal.*

- En la página Registro podemos escuchar las instrucciones de la página si presionamos el botón
- En la página Registro podemos registrar un usuario al presionar *Register* o por el comando de voz *registrar*.

Paso entre páginas al iniciar sesión.

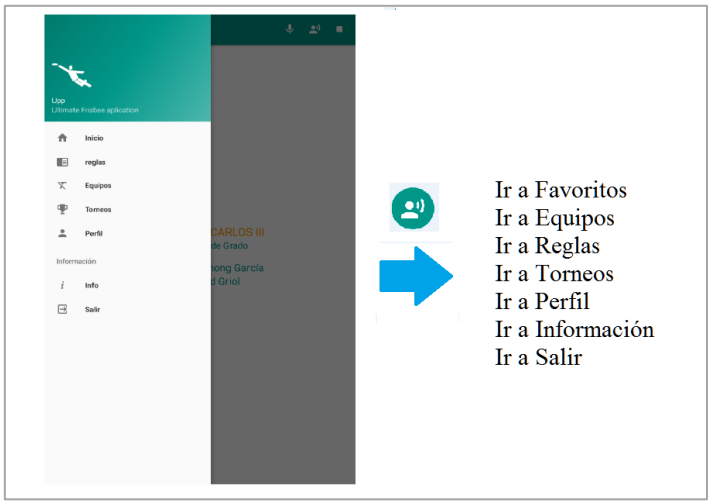


Figura 75: Flujo entre páginas

- Cuando iniciamos sesión en cualquier momento podemos saltar entre páginas o salir de aplicación, excepto cuando estamos viendo el contenido individual de un equipo, torneo o regla o cuando estamos en AR.
- Para pasar entre paginas presionamos en la barra superior el botón el cual desplegará el menú con las paginas donde deseamos ir, o podemos ir mediante el comando de voz *ir a* seguido del nombre de la página que deseamos ir.
- Para salir presionamos *salir* o decimos *salir*.

Favoritos

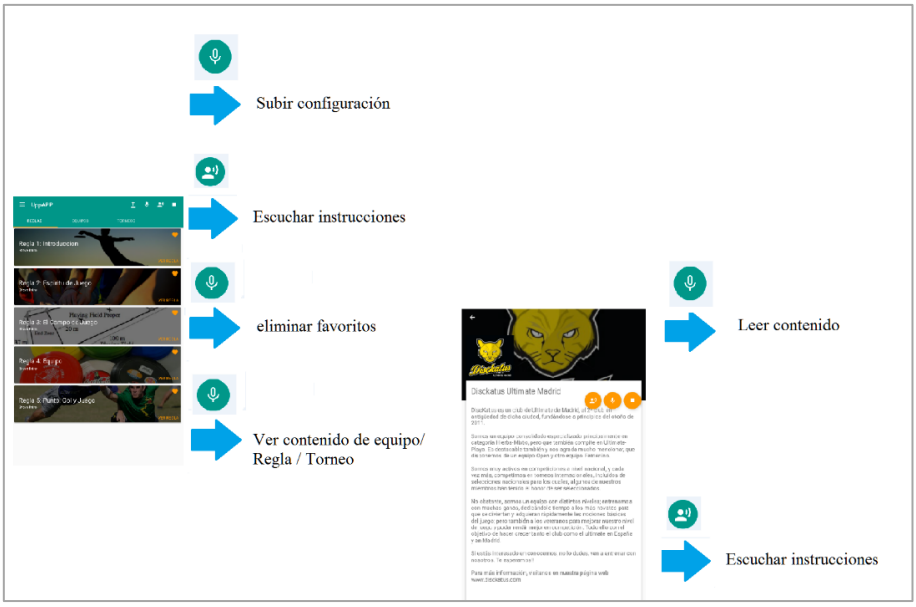


Figura 76: Flujo desde la página Favoritos

- En Favoritos podemos escuchar las instrucciones si pulsamos el botón .
- En la página Favoritos podemos subir la configuración actual de nuestras preferencias apretando el botón o por medio del comando de voz *subir*.
- En la página Favoritos podemos saltar entre listas de torneos, reglas y equipos presionando en las pestañas que se encuentran en la barra superior.
- En la página Favoritos podemos eliminar de la listas un elemento si presionamos el botón del elemento o a través del comando de voz *torneo/regla/equipo* + "el número asociado al elemento" + *no favorito*.
- En la página Favoritos podemos oír todos los torneos, reglas y equipos si ejecutamos los comandos de voz *oír torneos/reglas/equipos*.
- En Favoritos podemos seleccionar un elemento para iniciar la actividad que muestra su contenido apretando en el mismo elemento o mediante el comando de voz: *torneo/regla/equipo* + "el número asociado al elemento".
- En el contenido podemos leer el contenido mediante el comando de voz *leer*, podemos escuchar las instrucciones si apretamos el botón o podemos volver a la página de favoritos por el comando de voz *volver*.

Reglas

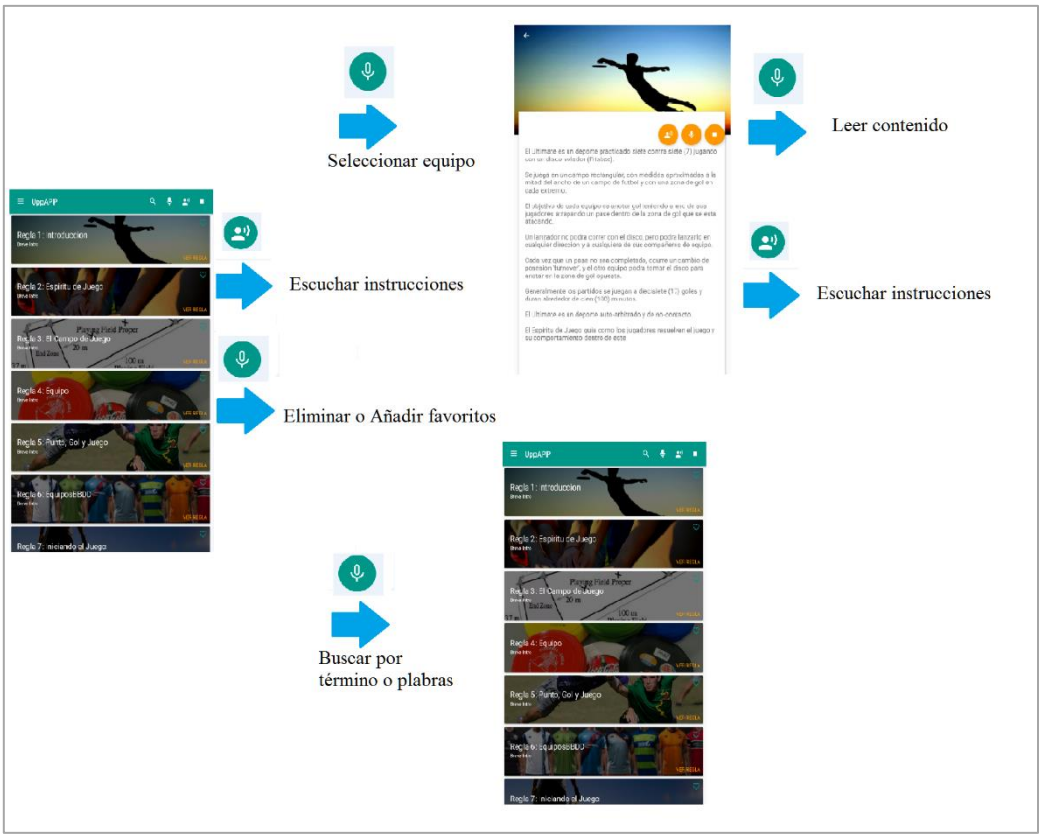


Figura 77: Flujo desde la página Reglas

- En Reglas podemos escuchar las instrucciones si presionamos el botón
- En la página Reglas podemos añadir una regla favorita si presionamos el botón del elemento o a través del comando de voz: *regla* + "el numero asociado a la regla" + *favorito*.
- En la página Reglas podemos eliminar una regla favorita si presionamos el botón del elemento o a través del comando de voz: *regla* + "el numero asociado a la regla" + *no favorito*.

- Podemos oír todas reglas si ejecutamos el comando de voz *oír reglas*.
- En la página Reglas podemos buscar elementos dentro de las reglas apretando en el botón , o por el comando de voz *buscar* + *frase/palabra*
- En la página Reglas podemos seleccionar una regla individual para iniciar la actividad que muestra su contenido apretando en el mismo elemento o mediante el comando de voz: *regla* + "el número asociado al elemento".
- En el contenido podemos leer el contenido mediante el comando de voz *leer*, podemos escuchar las instrucciones si apretamos el botón o podemos volver a la página de favoritos por el comando de voz *volver*.

Equipos

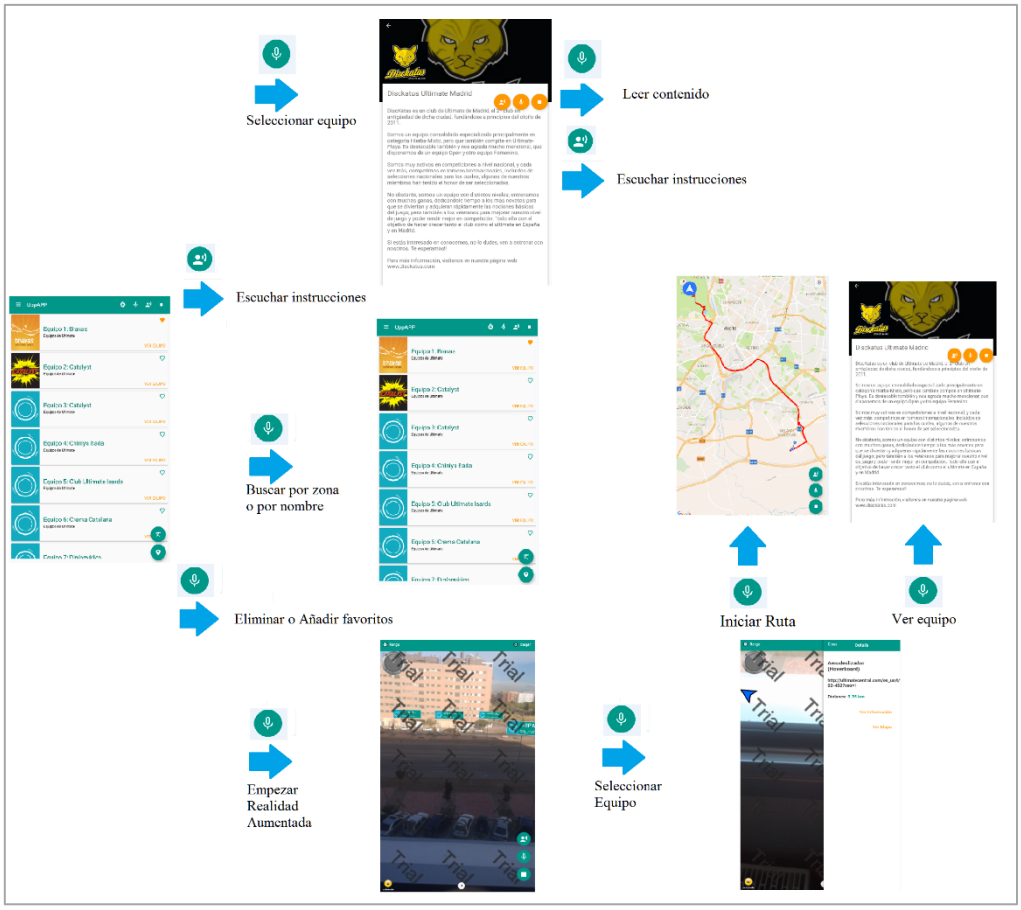


Figura 78: Flujo desde la página Equipos

- En Equipos podemos escuchar las instrucciones si presionamos el botón
- En la página Equipos podemos añadir una equipo favorito si presionamos el botón del elemento o a través del comando de voz: *equipo* + "el numero asociado al equipo" + *favorito*.
- En la página Equipos podemos eliminar un equipo favorito si presionamos el botón del elemento o a través del comando de voz: *equipo* + "el numero asociado al equipo" + *no favorito*.
- Podemos oír todos los equipos si ejecutamos el comando de voz *oír equipos*.
- Podemos buscar equipos por nombre apretando el botón , o por el comando de voz *buscar equipo* + *nombre del equipo.*

- Podemos buscar equipos por ubicación apretando el botón , o por el comando de voz *buscar zona* + *ubicación*.
- En la página Equipos podemos seleccionar un equipo individual para iniciar la actividad que muestra su contenido apretando en el mismo elemento o mediante el comando de voz: *equipo* + "el número asociado al elemento".
- En el contenido podemos leer el contenido mediante el comando de voz *leer*, podemos escuchar las instrucciones si apretamos el botón o podemos volver a la página de equipos por el comando de voz *volver*.
- Podemos iniciar la actividad de realidad aumentada si apretamos el botón , o por el comando de voz *realidad aumentada*.
- En la actividad realidad aumentada podemos seleccionar un equipo individual para iniciar la actividad que muestra su contenido seleccionado primero el elemento y luego apretando el botón *ver información*, o por medio del comando de voz: *equipo* + "el número asociado al equipo".
- En la actividad realidad aumentada podemos seleccionar un equipo individual para iniciar la actividad que muestra la ruta entre el equipo y el usuario, seleccionado primero el elemento y luego apretando el botón *ver Mapa*, o por medio del comando de voz: *equipo* + "el número asociado al equipo" + *ruta*.
- En la actividad ruta podemos iniciar la aplicación Google Maps que muestra las indicaciones de la ruta por medio del botón o por el comando de voz: *iniciar ruta*.

Torneos

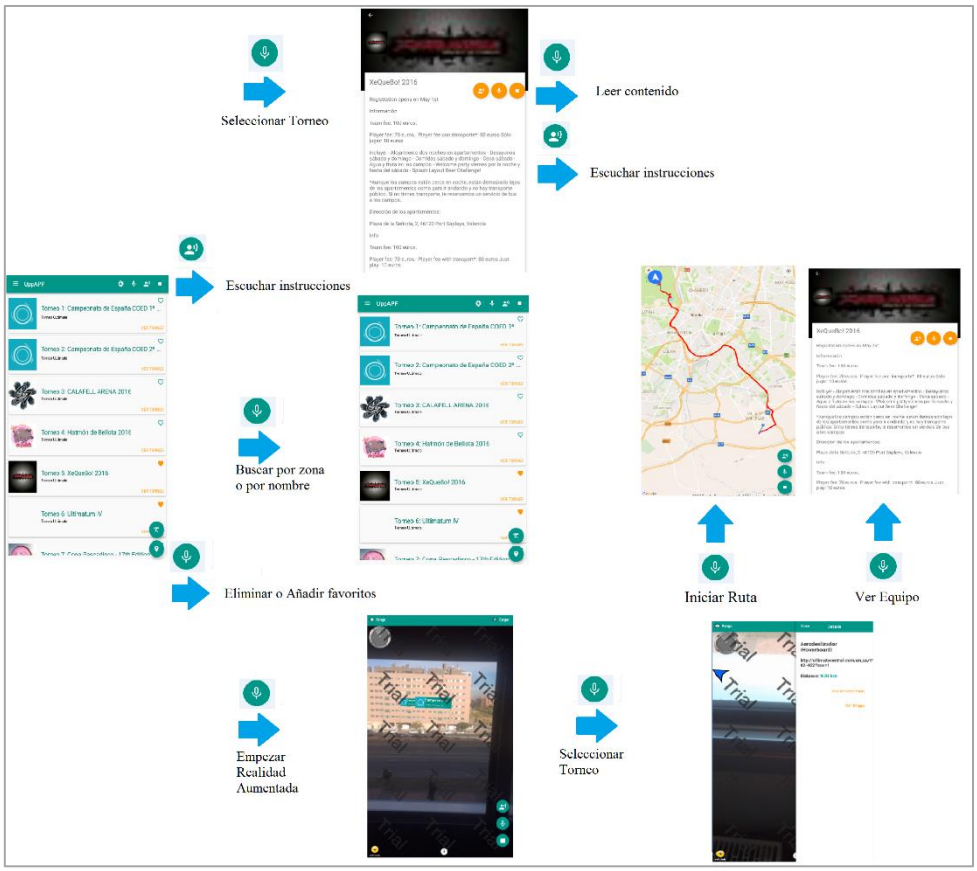


Figura 79: Flujo desde la página Torneos

- En Torneos podemos escuchar las instrucciones si presionamos el botón
- En la página Torneos podemos añadir una torneo favorito si presionamos el botón del elemento o a través del comando de voz: *torneo* + "el numero asociado al torneo" + *favorito*.
- En la página Torneos podemos eliminar un torneo favorito si presionamos el botón del elemento o a través del comando de voz: *torneo* + "el numero asociado al torneo" + *no favorito*.
- Podemos oír todos los torneos si ejecutamos el comando de voz *oír torneos*.
- Podemos buscar torneos por nombre apretando el botón , o por el comando de voz *buscar torneos* + *nombre del torneo.*
- Podemos buscar torneos por ubicación apretando el botón , o por el comando de voz *buscar zona* + *ubicación* .
- En la página Torneos podemos seleccionar un torneo individual para iniciar la actividad que muestra su contenido apretando en el mismo elemento o mediante el comando de voz: *torneo* + "el número asociado al elemento".
- En el contenido podemos leer el contenido mediante el comando de voz *leer*, podemos escuchar las instrucciones si apretamos el botón o podemos volver a la página de torneos por el comando de voz *volver.*
- Podemos iniciar la actividad de realidad aumentada si apretamos el botón , o por el comando de voz *realidad aumentada*.
- En la actividad realidad aumentada podemos seleccionar un torneo apretando en el elemento mismo.
- En la actividad realidad aumentada podemos seleccionar un torneo individual para iniciar la actividad que muestra su contenido seleccionado primero el elemento y luego apretando el botón *ver información*, o por medio del comando de voz: *torneo* + "el número asociado al torneo".
- En la actividad realidad aumentada podemos seleccionar un torneo individual para iniciar la actividad que muestra la ruta entre el torneo y el usuario, seleccionado primero el elemento y luego apretando el botón *ver Mapa*, o por medio del comando de voz: *torneo* +"el número asociado al torneo" + *ruta*.
- En la actividad ruta podemos iniciar la aplicación Google maps que muestra las indicaciones de la ruta por medio del botón o con el comando de voz: iniciar ruta.

Capítulo 5

Evaluación

En este capítulo se recogen las evaluaciones realizadas por parte de los usuarios. Las impresiones de los usuarios se registran a través de un formulario entregado después de utilizar la aplicación. Además se analizan los resultados obtenidos, y se muestran las conclusiones derivadas del análisis.

5.1 Formulario

El formulario consta de 16 preguntas referentes al conocimiento del deporte, al uso de dispositivos móviles y sus tecnologías, al uso de estas últimas en la aplicación, y preguntas relacionadas a la aplicación en general.

Número	Pregunta	Rango
1	¿Tiene experiencia en el deporte Ultimate Frisbee?	1:Nada / 5:Mucha
2	¿Usa habitualmente aplicaciones móviles?	1:Nada / 5:Mucha
3	¿Suele utilizar la salida voz del dispositivo para la lectura y acceso al contenido?	1:Nada / 5:Mucha
4	¿Suele utilizar la entrada voz para interaccionar con el dispositivo?	1:Nada / 5:Mucha
5	¿Está familiarizado con la Realidad Aumentada?	1:Nada / 5:Mucha
6	¿Ha utilizado aplicaciones con Realidad Aumentada?	1:Nada / 5:Mucha
7	¿Entiende bien los mensajes de voz del sistema?	1:Nada / 5:Mucha
8	¿Las instrucciones del programa son claras y entendibles?	1:Nada / 5:Mucha
9	¿Los comandos por voz son bien respondidos?	1:Nada / 5:Mucha
10	¿Le ha parecido útil la realidad aumentada en la aplicación?	1:Nada / 5:Mucha
11	¿Se muestran los puntos de interés de forma correcta?	1:Nada / 5:Mucha
12	¿El ritmo de interacción es fluido?	1:Nada / 5:Mucha
13	¿Ha entendido en cada instante qué le requería la aplicación hacer a continuación?	1:Nada / 5:Mucha
14	¿La interfaz es intuitiva?	1:Nada / 5:Mucha
15	¿Le ha parecido útil la aplicación?	1:Nada / 5:Mucha
16	¿Qué nota global le da a la aplicación?	1:Muy Mala / 5:Muy buena

Tabla 11: Formulario para la evaluación de la aplicación

5.2 Respuestas

Las repuestas son obtenidas de 15 usuarios diferentes, de los cuales 10 de ellos son jugadores o han practicado Ultimate Frisbee y 5 son usuarios que no están relacionados con el deporte.

Persona	**Preguntas**															
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
1	3	5	1	1	2	2	3	3	3	4	4	5	4	3	4	4
2	4	4	1	1	2	2	5	3	4	4	4	4	4	4	5	4
3	3	4	2	2	1	1	4	3	3	4	3	3	4	3	5	4
4	4	4	1	2	2	2	5	4	4	4	3	5	4	4	4	3
5	4	5	1	2	2	1	4	3	4	3	4	4	4	3	5	4
6	4	5	2	2	3	2	4	3	4	4	4	3	5	4	4	4
7	5	5	1	2	1	1	3	2	3	5	3	4	4	3	5	4
8	5	5	1	1	1	1	5	3	3	5	4	4	4	4	4	3
9	4	5	1	1	2	2	4	3	2	3	4	5	4	4	5	4
10	3	3	1	1	1	1	3	3	3	2	4	3	3	3	3	3
11	0	5	2	2	2	2	4	3	3	3	3	4	5	3	5	4
12	0	5	1	1	2	1	4	4	2	3	3	3	4	3	4	3
13	0	5	0	1	1	1	3	3	4	2	4	4	5	3	3	4
14	0	5	1	2	1	1	3	3	3	3	4	5	4	4	3	3
15	0	5	1	2	3	3	5	3	3	3	3	3	4	4	3	4
Media	**2,6**	**4,7**	**1,1**	**1,5**	**1,7**	**1,5**	**3,9**	**3,1**	**3,2**	**3,5**	**3,6**	**3,9**	**4,1**	**3,5**	**4,1**	**3,7**
Varianza	**4**	**0,4**	**0,3**	**0,3**	**0,5**	**0,4**	**0,6**	**0,2**	**0,5**	**0,8**	**0,3**	**0,6**	**0,3**	**0,3**	**0,7**	**0,2**
Mediana	**3**	**5**	**1**	**2**	**2**	**1**	**4**	**3**	**3**	**3**	**4**	**4**	**4**	**3**	**4**	**4**

Tabla 12: Respuestas obtenidas por parte de los usuarios

5.3 Conclusiones extraídas

En primer lugar vamos a extraer las conclusiones que indistintamente del tipo de usuario que sea, tienen un calado general en la aplicación.

Por ejemplo, podemos concluir que las personas que han realizado la encuesta utilizan frecuentemente un dispositivo móvil, pero tienen poca o ninguna experiencia con las tecnologías de síntesis y reconocimiento de voz, y con tecnologías de realidad aumentada; por lo se han enfrentado a una aplicación con recursos que normalmente no utilizan, lo que hace una buena referencia a la hora de saber si la aplicación es fácil e intuitiva de usar.

Por otra parte, los usuarios que han utilizado la aplicación han entendido bien la síntesis de voz, aunque las instrucciones no han sido del todo claras, o han sido un tanto confusas en algunos casos. De este punto podemos también afirmar que el reconocimiento a los mandatos por voz del usuario funciona correctamente, pero aún no son los mejores posibles.

Con respecto a la realidad aumentada, vemos que los usuarios que practican el deporte les parece bastante útil esta herramienta con una casi una uniformidad en las respuestas. Sin embargo, con los otros tipos de usuario la utilidad de la herramienta baja un punto.

Pasamos los resultados que hacen referencia la visión general de la aplicación. En este aspecto podemos observar que en las repuestas obtenidos a las preguntas relacionadas a este punto, la media y mediana tienen un resultado satisfactorio, estando o llegando casi al 4, y con un valor de varianza bastante bajo, lo que nos hace concluir que la aplicación es una aplicación fluida, relativamente fácil de entender, intuitiva, y con la que se tienen una buena experiencia, aunque es posible mejorar algunos aspectos de la misma.

Capítulo 6

Conclusiones y trabajo futuro

En este capítulo detallamos las posibles mejoras que puede tener la aplicación una vez finalizada la misma, además de las conclusiones finales del proyecto.

6.1 Conclusiones finales

En primer lugar el desarrollo de la aplicación se realizó en el contexto del Ultimate Frisbee, y aunque existen muchos tipos de aplicaciones de deporte, no existen muchas específicas que además añadan las funciones de RA y síntesis y reconocimiento de voz, por lo que en este marco y teniendo en cuenta los resultados obtenidos por el formulario, podemos decir que la satisfacción de alcanzar una aplicación que llega a ser útil es bastante alta.

Por otra parte nos encontramos el problema de recurrir a tecnologías que no se suelen utilizar, lo que conlleva la búsqueda e investigación de recursos para poder realizarlo de forma satisfactoria. El estudio previo de las bibliotecas de síntesis y reconocimiento de voz, y después el estudio de las tecnologías de realidad aumentada, aumento la fascinación por estas tecnologías y supuso un reto a la hora de alcanzar la meta final.

Hemos comprobado además que cada día es más popular este tipo de tecnología y utilizarla en el proyecto supuso diseñar una interfaz amigable que a la vez sea útil e intuitiva. Todos estos problemas han ayudado a alcanzar unos conocimientos fuertes acerca de la implantación de estas tecnologías en dispositivos móviles.

En este contexto podemos afirmar que se ha alcanzado un conocimiento bastante profundo de cómo debe ser una aplicación móvil con Android como sistema operativo, que consultas y librerías podemos hacer en este tipo de aplicaciones y cuál es la mejor alternativa según nuestras necesidades. Por ejemplo la elección de utilizar Volley o Retrofit para consultar y comunicarse elementos externos. Al final se optó por utilizar ambas ya que tenían mejores prestaciones en diferentes ámbitos.

Por otra parte nos hemos enfrentado al problema de la lectura de la página web para obtener los datos. Este apartado ha sido especialmente difícil de diseñar, ya que por una parte queríamos tener una base de datos propia, y a la vez comunicarnos con una página

externa. La página además tenía la dificultad de no poseer un API muy desarrollado por lo que solo nos quedaba la opción del *Scraping* de la página. En este sentido fue complicado diseñar la carga de los elementos individuales, ya que los elementos de la página variaban mucho según qué resultado se obtenía, es decir, no siempre las etiquetas de la página eran las mismas, aun siendo la consulta de la misma naturaleza.

Otro punto importante fue la carga de los POI en RA. Los POI necesitan longitudes y latitudes para mostrarse y nuestra aplicación debía ser capaz de proporcionar estos datos al sistema. El problema surgió debido a que la página consultada no contaba con los datos en el *scraping* ni con una base de datos a la que comunicarse. Como solución se optó por crear los propios equipos y torneos cuando se cargaban las preferencias del usuario, para luego añadir los datos manualmente. De esta forma se paleaba el problema, aunque se pretende en un futuro mejorar este aspecto.

En conclusión la utilización primero de tecnologías novedosas como las síntesis, reconocimiento de voz y la realidad aumentada, unida al uso de muchos tipos de librerías para resolver los problemas que surgieron a la hora de diseñar la aplicación ha ayudado a tener una visión y un conocimiento amplio del sistema Android, añadir funcionalidades al propio sistema, y por último que el autor consiga versatilidad a la hora de conseguir soluciones para muchos tipos de problemas diferentes.

6.2 Futuras mejoras

Una vez terminada la aplicación y observando las conclusiones obtenidas podemos indicar una serie de mejoras que podrían incrementar las funcionalidades de la aplicación.

1. Añadir automáticamente las longitudes y latitudes. Esta opción no está disponible aún, ya que la página WEB no cuenta con ello, y la API de la misma está en proceso de mejora.

2. Si tenemos la primera mejora se puede cargar los equipos o torneos en RA sin necesidad de comunicarse con el servidor externo, ya que solo haría falta saber el nombre de la ubicación, hacer el WEB SCRAPING, y visualizar los torneos o equipos que devuelve la acción.

3. Crear un servicio Cloud para la modificación automática de las preferencias.

4. Añadir imágenes de los equipos y torneos en los markers en la RA. Aunque esta opción puede incrementar el peso de la aplicación si se guarda en la misma, y ralentizar la carga.

5. Mejorar las preguntas por voz detectadas por la aplicación, de esta forma el usuario puede interactuar mejor con ella.

6. Añadir una pestaña de Jugadores para así poder ver a los jugadores individualmente.

7. Añadir la funcionalidad de poder visualizar videos cuando estamos en la pantalla de realidad aumentada, ya que Wikitude permite esta opción.

8. Poder visualizar todos los puntos de interés en un mapa sin utilizar RA.

9. Con Wikitude podemos leer una imagen con la cámara del dispositivo y crear una acción, de esta forma una posible funcionalidad para el futuro será leer el logo de un equipo y abrir automáticamente la página del mismo.

10. Portar el proyecto a otras plataformas como iOS o Windows Phone, analizando los posibles problemas que pueden surgir.

Presupuesto

En este apartado se detalla el presupuesto de nuestra aplicación: los gastos de personal, software y hardware generados a lo largo del desarrollo del proyecto.

En el calendario planificado se especifica que la jornada laboral será de 3 horas al día de lunes a jueves, lo que supone que la carga de trabajo será continua durante toda la vida del proyecto.

La carga aunque ha sufrido irregularidades no afecta en suma a lo planificado anteriormente por lo que nos ceñiremos a esta planificación para calcular los costes.

Coste personal

Fase	Duración /días	Horas / día	Coste hora/ingeniero €	Total (€)
Planificación	67	3	30	6.030
Ejecución	77	3	30	6.930
Documentación	35	3	30	3.150
				Total: 16.110 €

Tabla 13: Coste Personal

Coste Sofware

Software	Coste
Microsoft Office 2013	119
Adobe Photoshop	120
Adobe Illustrator	120
	Total: 359 €

Tabla 14: Coste software

Coste Hardware

Descripción	Coste (€)	% uso dedicado	Dedicación meses	Periodo de depreciación	Total (€)*
HP 110-526nf i3	500	100	10	60	83,3
ASUS Pro	550	100	10	60	91,7
Nvidia Shield	240	100	10	60	40
HTC One	120	100	10	60	20
Cable usb	5	100	10	60	0,9
					Total: 235,9 €

Tabla 15: Amortización Hardware

* La fórmula seguida es: (A /B)xCxD [42]

A = número de meses desde la fecha de facturación en que el equipo es utilizado
B = periodo de depreciación (60 meses)
C = coste del equipo (sin IVA)
D =% del uso que se dedica al proyecto

Resumen

Descripción	Coste Total €
Personal	16.110
Amortización Hardware	235,9
Software	359
Costes indirectos (20%)	3340,98
	Total sin IVA: 20045,88€
	Total con IVA (18%): 23654,14€

Tabla 16: Coste total

El coste final del proyecto asciende a 23654,14€.

Glosario

Android Studio o Android SDK: entorno de desarrollo Android.

Android: sistema operativo diseñado principalmente para dispositivos móviles.

API (Application Programming Interface): es el conjunto de subrutinas, funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

CSS: lenguaje usado para definir y crear la presentación de un documento estructurado escrito en HTML o XML.

Diagramas de Gantt: herramienta gráfica cuyo objetivo es exponer el tiempo de dedicación previsto para diferentes tareas o actividades a lo largo de un tiempo total determinado.

Framework: conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar.

Google: compañía especializada en productos y servicios relacionados con Internet, software, dispositivos electrónicos y otras tecnologías.

Hardware: conjunto de elementos físicos o materiales que constituyen una computadora o un sistema informático.

Hosting: servicio que provee a los usuarios de Internet un sistema para poder almacenar información, imágenes, vídeo, o cualquier contenido accesible vía web.

HTML: lenguaje de marcado para la elaboración de páginas web.

HTTP (Hypertext Transfer Protocol): protocolo de comunicación que permite las transferencias de información en la World Wide Web.

Java: lenguaje de programación de propósito general, concurrente, orientado a objetos.

JavaScript o JS: lenguaje de programación web.

JQuery: Librería JavaScript para desarrollo de páginas web.

JSON: formato de texto ligero para el intercambio de datos.

Lenguaje C: Lenguaje de programación.

Linux: sistema operativo de desarrollo libre.

MySQL: sistema de gestión de bases de datos relacional.

PHP: lenguaje de programación de uso general de código del lado del servidor.

POI (Point of interest): puntos de interés en RA.

Realidad aumentada (AR) (RA): visión a través de un dispositivo tecnológico, directa o indirecta, de un entorno físico del mundo real.

REST: estilo de arquitectura software para sistemas hipermedia distribuidos como la World Wide Web.

Smartphones: tipo de teléfono móvil construido sobre una plataforma informática móvil.

Software: conjunto de programas y rutinas que permiten a la computadora realizar determinadas tareas.

SQLite: base de datos relacional para dispositivos Android.

Tablets: ordenador portátil de mayor tamaño que un teléfono inteligente.

Ultimate Frisbee: es un deporte competitivo de equipo sin contacto entre jugadores.

URL: identificador de recursos uniforme.

Web Scraping: técnica utilizada mediante programas de software para extraer información de sitios web.

WIFI: mecanismo de conexión de dispositivos electrónicos de forma inalámbrica.

Work BreakdownStructure (WBS): paradigma organizativo de proyectos informáticos.

World Wide Web: sistema de distribución de documentos de hipertexto o hipermedios interconectados y accesibles vía Internet.

XML: meta-lenguaje que permite definir lenguajes de marcas.

Anexos

Encuesta preparatoria

Encuesta realizada para la creación de la aplicación:

	C	D	E	F	G	H	I	J	K
1	Edad	Años Jugando	Equipo en el que juega	Posición	Como sientes el ultimate	¿Crees que ex	Que tipo de Aplicaciones	¿Sueles utilizar aplicacio	Si la respuesta anterior e
2	27	4	Diskatus	Handler/Cutter	9	No	Nike	No	
3	30	10 meses	Diskatus	Cutter	10	NS/NC		No	
4	25	2	diskatus	Handler/Cutter	8	Si	nada	Si	
5	31	0	Diskatus	Cutter	9	NS/NC	Medidores de actividad, t	No	
6	23	1	Diskatus	Cutter	7	NS/NC	Ninguna	No	
7	35	4	Diskatus / Mubidisk	Cutter	9	Si	ninguna	Si	la mia de las reglas jeje
8	24	7	Diskatus	Handler	7	No	ninguna	De vez en cuando	
9	25	2	Diskatus	Handler/Cutter	9	NS/NC		No	
10	22	2	Diskatus	Cutter	8	NS/NC	ninguna	De vez en cuando	Reglas Taty
11	25	5	Diskatus	Handler	8	No		No	
12	26	6 meses	Diskatus	Cutter	9	NS/NC		No	
13	30	4	Diskatus	Cutter	8	NS/NC	No	No	
14	36	4 y medio	Diskatus	Handler	8	NS/NC	Google Tracks, Reglas U	Si	Reglas Ultimate, Ultimate
15	33	1	Diskatus	Cutter	9	Si	Ninguna	No	

	L	M
1	¿Que buscarías en una aplicación de ultimate?	Mostrar la información
2	Reglas, Jugadas de Partido / Pases, Calentamiento / Drills de calentamiento, Torneos, Crear Jugadas	Texto, Videos, Fotos
3	Reglas, Jugadas de Partido / Pases, Calentamiento / Drills de calentamiento, Hacer rutinas de calentamiento, Crear Jugadas, Seguimiento de jugadores (pases conseguido, cortes realizados, etc)	Texto, Videos, Fotos
4	Calentamiento / Drills de calentamiento, Equipos, Torneos, Crear Jugadas	Texto, Fotos
5	Reglas, Jugadas de Partido / Pases, Calentamiento / Drills de calentamiento, Equipos, Torneos, Hats, Hacer rutinas de calentamiento	Texto, Videos, Fotos
6	Reglas, Jugadas de Partido / Pases, Equipos, Torneos, Hats, Crear Jugadas	Texto, Videos, Fotos
7	Reglas, Jugadas de Partido / Pases, Torneos, Crear Jugadas	Texto, Videos, Fotos
8	Reglas, Jugadas de Partido / Pases, Calentamiento / Drills de calentamiento, Hacer rutinas de calentamiento, Seguimiento de jugadores (pases conseguido, cortes realizados, etc)	Texto, Videos, Audio,
9	Reglas, Jugadas de Partido / Pases, Crear Jugadas, Seguimiento de jugadores (pases conseguido, cortes realizados, etc)	Texto, Fotos
10	Reglas, Jugadas de Partido / Pases, Equipos, Torneos, Crear Jugadas	Texto, Videos, Fotos
11	Jugadas de Partido / Pases, Calentamiento / Drills de calentamiento, Torneos, Hacer rutinas de calentamiento, Crear Jugadas, Seguimiento de jugadores (pases conseguido, cortes realizados, etc)	Texto, Videos, Fotos
12	Calentamiento / Drills de calentamiento, Equipos, Torneos, Hacer rutinas de calentamiento, Seguimiento de jugadores (pases conseguido, cortes realizados, etc)	Videos, Fotos
13	Reglas, Jugadas de Partido / Pases, Calentamiento / Drills de calentamiento, Equipos, Torneos, Hats, Hacer rutinas de calentamiento, Seguimiento de jugadores (pases conseguido, cortes realizados, et	Texto, Videos, Fotos
14	Reglas, Calentamiento / Drills de calentamiento, Crear Jugadas, Seguimiento de jugadores (pases conseguido, cortes realizados, etc), A mi me gustaría una app para estudiar ultimate: reglas, drills para	Texto, Fotos
15	Reglas, Equipos, Torneos, Hats	Texto, Videos, Fotos,

Bibliografía

[1] Manghi Haquin, Dominique. La perspectiva multimodal sobre la comunicación. Desafíos y aportes para la enseñanza en el aula. *Diálogos Educativos* [en línea]. 17 de Enero 2012, vol. 11. [Consulta: diciembre 2015]. http://revistas.umce.cl/dialogoseducativos/article/viewFile/102/110

[2] Kress, G. & van Leeuwen, T. (2001). *Multimodal Discourse - The Modes and Media of Contemporary Communication*. Londres: Arnold.

[3] Procesamiento del Habla e Interacción Multimodal. Consulta en Diciembre de 2015. http://www.ugr.es/~rlopezc/phim.htm

[4] Building more accessible technology. Consulta en Enero de 2016. https://googleblog.blogspot.com.es/2016/04/building-more-accessible-technology.html

[5] Fundación Telefónica (2011). *Realidad aumentada: una nueva lente para ver el mundo* (1ra edición). España. Editorial: Ariel.

[6] Todo lo que necesitas saber de Pokémon. Consulta en Enero de 2016. http://www.bbc.com/mundo/noticias-36785172

[7] TripAdvisor adds augmented reality to mobile and iPad apps. Consulta en Enero de 2016. https://www.tnooz.com/article/tripadvisor-adds-augmented-reality-to-mobile-and-ipad-apps/

[8] Spirit of the game. Consulta Noviembre de 2015. http://www.usaultimate.org/spirit/

[9] Aplicaciones Deportivas. Consulta Noviembre de 2015. https://play.google.com/store/search?q=deportes

[10] Ableson, W. Frank; Collins, Charles; Sen, Robi. (2010). *Android: guía para desarrolladores*. Madrid, España. Editorial Anaya Multimedia.

[11] iOS y Android copan casi el 100% de la cuota de mercado. Consulta Noviembre de 2015. http://computerhoy.com/noticias/moviles/ios-android-copan-casi-100-cuota-mercado-49794

[12] Android en 2016: KitKat a la cabeza; Marshmallow, rezagado. Consulta en Abril de 2016. https://www.cnet.com/es/noticias/distribucion-android-2016-kitkat-marshmallow/

[13] API de Android 5.0 Consulta en Noviembre de 2015. https://developer.android.com/about/versions/android-5.0.html?hl=es

[14] Novedades más importantes que nos ofrece Android 5.0 Lollipop. Consulta en Noviembre de 2016. http://andro4all.com/2014/11/novedades-android-5-0-lollipop

[15] Ribas Lequerica, Joan (2014). *Desarrollo de aplicaciones para Android*. Madrid, España. Editorial Anaya Multimedia.

[16] Android Studio. Consulta en Noviembre de 2015. https://developer.android.com/studio/index.html

[17] ¿Qué es XAMPP? Consulta en Noviembre de 2015. https://www.apachefriends.org/es/index.html

[18] AppServ: Apache + PHP + MYSQL. Consulta en Noviembre de 2015. https://www.appserv.org/en/

[19] Buyya, Rajkumar; Broberg, James; Goscinski, Andrzej (2011). *Cloud Computing: Principles and Paradigms* (pages 3–9). New Jersey, United States of America.

[20] MariaDB versus MySQL - Caracteristicas. Consulta en Febrero de 2016. https://mariadb.com/kb/es/mariadb-versus-mysql-features/

[21] Ultimate Central. Consulta en Enero de 2016. https://ultimatecentral.com

[22] Parsing HTML in Android with Jsoup. Consulta en Marzo de 2016. http://www.survivingwithandroid.com/2014/04/parsing-html-in-android-with-jsoup-2.html

[23] Transmitting Network Data Using Volley. Consulta en Marzo de 2016. https://developer.android.com/training/volley/index.html

[24] Android Working with Retrofit HTTP Library. Consulta en Marzo de 2016. http://www.androidhive.info/2016/05/android-working-with-retrofit-http-library/

[25] Parsing HTML in Android with Jsoup. Consulta en Marzo de 2016. http://www.survivingwithandroid.com/2014/04/parsing-html-in-android-with-jsoup-2.html

[26] Introducing JSON. Consulta en Noviembre de 2015. http://www.json.org/

[27] XML ¿QUÉ ES? Consulta en Noviembre de 2015. http://www.mundolinux.info/que-es-xml.htm

[28] Material design. Consulta en Noviembre de 2015. https://material.google.com/

[29] Williams, Jason D. (2009). *Spoken dialogue systems: challenges, and opportunities for research* [PDF file]. Recuperado de https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/williams-2009-asru.pdf

[30] Ramon Lopez Cozar Delgado, Masahiro Araki (2005). *Spoken, Multilingual and Multimodal Dialogue Systems: Development and Assessment* (pages 229–231). DOI: 10.1002/0470021578

[31] Google Assistant, el asistente de voz que llegará a Android. Consulta en Enero de 2016. http://www.dispositivomovil.com/google-assistant-el-asistente-de-voz-que-llegara-a-android/

[32] Siri. Consulta en Enero de 2016. http://www.apple.com/es/ios/siri/

[33] Join the Ivona Text-to-Speech team. Consulta en Enero de 2016. https://www.ivona.com/

[34] 10 Sintetizadores de voz (TTS) para Android. Consulta en Diciembre de 2015. http://www.emezeta.com/articulos/10-sintetizadores-de-voz-tts-para-android

[35] Android Speech. Consulta en Diciembre de 2015. https://developer.android.com/reference/android/speech/package-summary.html

[36] ¿Qué es la realidad aumentada? Consulta en Febrero de 2016. http://realidadaumentada.info/tecnologia/

[37] Definición de Realidad Aumentada. Consulta en Febrero de 2016. http://wwwavancesdelcelular.weebly.com/definicion.html

[38] Documentation wikitude sdk android. Consulta en Enero de 2016. http://www.wikitude.com/developer/documentation/android

[39] Preferencias en Android I: Shared Preferences. Consulta en Noviembre de 2015. http://www.sgoliver.net/blog/preferencias-en-android-i-shared-preferences/

[40] Lo mejor de Google Maps para cada aplicación de Android. Consulta en Enero de 2015. https://developers.google.com/maps/documentation/android-api/?hl=es

[41] Google Maps Draw Route between two points using Google Directions in Google Map Android API V2. Consulta en Marzo de 2016. http://www.androidtutorialpoint.com/intermediate/google-maps-draw-path-two-points-using-google-directions-google-map-android-api-v2/

[42] Proyecto fin de carrera. Consulta en Agosto de 2016. http://www.uc3m.es/ss/Satellite/SecretariaVirtual/es/TextoMixta/1371218569460/Proyecto_fin_de_carrera#publicidaddelamemoria